Aicumene:2026.040v10 [cs.AI] October 2026

Verifiable AI for Law: a legal verifiability compiler with a rule/judgment firewall

Architecture, conditional guarantees, the common-source floor, and a pre-registrable evaluation protocol

Alexandra Bernadotte
Aicumene
bernadotte@aicumene.com

Ivan Menshikov
Aicumene

October 2026 · Aicumene:2026.040v10 · first version June 2026

Abstract

Legal language-model agents fabricate authorities, overstate discretionary conclusions, and present incomplete analyses as final; audits of general-purpose models and of commercial retrieval-augmented legal research tools measure these failures at rates that no professional standard tolerates. This paper presents *Verifiable AI for Law*, a bounded decision-support architecture centered on an independent deterministic *verifiability compiler*. Proposer and Critic agents generate and challenge a typed *Application-IR*; the compiler checks citations, jurisdiction, source currency and formally decidable predicates against an approved executable rulebook and validated case-specific facts, producing auditable diagnostics. A *rule/judgment firewall* separates mechanically verifiable conditions from open-textured legal standards and escalates the latter to qualified human reviewers; distinct authority and evidence retrieval paths retain provenance; capability-scoped execution isolates matters; and completion of a long-horizon matter is a verified state, not a model's confidence. We state what the architecture guarantees and what it cannot. Soundness is conditional on correctly encoded, current authority and authenticated facts, and we make that condition quantitative: if a share c of a proposer's errors originates in a source the compiler also reads, no compiler configuration reduces the false-accept rate below $p\,c$, where p is the proposer's error rate. A reproducible control-layer experiment with a fallible verifier confirms the closed form $\mathrm{FA} = \frac{1}{2}[p\,c + (1 - p\,c)\,p_{\mathrm{rule}}]$: a verifier that shares half the proposer's sources with perfect rule encoding is worse than one with independent sources and a five percent encoding error. Source governance is therefore a first-order control, and agreement between language models drawn from one source is not verification. We describe the prototype, two illustrative workflows, the security design, a staged rollout, and a pre-registrable factorial evaluation protocol with five falsifiable hypotheses. The system does not establish the legal correctness of its rulebook and does not replace professional judgment. The model proposes; the compiler disposes.

Keywords: verifiable AI; legal AI; deterministic compiler; rule/judgment firewall; Application-IR; executable law; legal retrieval; provenance; common-source error; agent security; professional responsibility.
JEL: K10, K40, O33, C63.
Status: Aicumene Research preprint, v10, prepared for SSRN. First version June 2026, ResearchGate doi:10.13140/RG.2.2.18174.75841.
Online: https://research.aicumene.com/papers/verifiable-law/.

1 Why raw LLM outputs are insufficient in law

Two recurrent failure modes matter: **fabrication / over-assertion** (invented citations, or discretionary standards such as "extreme hardship"[1] and "well-founded fear"[2] treated as mechanically settled) and **premature completion** (partial analyses presented as final). These are not anecdotes. On verifiable questions about United States federal case law, general-purpose language models hallucinate on a majority of queries, worse for lower courts and older cases [19]; commercial retrieval-augmented legal research products from the two largest providers still produce unsupported or mis-grounded answers in roughly one query in six to one in three [23]. Merely scaling a model does not make its output independently auditable. The broader epistemic problem of establishing structurally measurable access to truth is discussed in the April Theses of the New Cognitive Paradigm [7]. A legal decision-support system must expose what can be verified, what cannot, and why.

Contributions. The paper makes five contributions. (1) An architecture, *Verifiable AI for Law*, in which a deterministic compiler, not a second language model, disposes of formally decidable claims, with a typed Application-IR, source-linked diagnostics, a rule/judgment firewall, dual authority-and-evidence retrieval, and verified stopping (Sections 3–11). (2) A precise statement of what the compiler guarantees: conditional soundness, verification coverage, and a *common-source floor* on false acceptance that no verifier reading the proposer's sources can lower (Section 12). (3) A reproducible control-layer experiment with a fallible verifier that confirms the closed form of the floor and shows that source independence dominates the verifier's own accuracy (Section 13). (4) A formal basis for the isolation axis, guarded forward invariance and observation indistinguishability, specialized to a tool-using legal agent (Section 11). (5) A pre-registrable factorial evaluation protocol with five falsifiable hypotheses (Section 15). What the paper does *not* claim is stated in Section 16: no benchmark score on real legal matters, no proof that any rulebook is legally correct, no autonomous legal advice.

2 Related work

Computational law and executable rules. The idea that statutory rules can be executed by a machine is four decades old: Sergot and colleagues formalized the British Nationality Act as a logic program and showed both the power and the limits of the exercise [26]; Genesereth's programme of computational law asks for machine-readable rules that a regulator can run [20]; Lawsky argued that a non-monotonic logic is needed to capture the default-and-exception structure of statutes [21]; and Catala is a programming language designed to translate statutory text into executable code faithfully, with lawyers in the loop and a formal semantics [25]. The executable rulebook of our architecture belongs to this tradition, and its rules point back to Akoma Ntoso texts, the OASIS standard for machine-readable legal documents [4]. The compiler presupposes that a rulebook exists and is source-linked; how faithfully a rule was formalized is the question Catala addresses, which is why source-to-rule correspondence is a first-order control here rather than a detail.

Hallucination in legal AI. Dahl et al. profiled legal hallucinations in large language models and found them pervasive and structured [19]; Magesh et al. extended the audit, in a pre-registered design, to retrieval-augmented commercial products and found that retrieval reduces but does not remove unsupported or mis-grounded answers [23]. These measurements motivate the two failure modes of Section 1 and the choice to delegate decidable claims to a non-generative oracle rather than to a second model.

Judging models with models. Agreement between language models is widely used as a proxy for correctness, and the limits of that proxy are documented [29]. Our Proposer–Critic pair follows this practice

[1] *Extreme hardship* is a context-sensitive standard used in certain U.S. immigration waivers. It requires hardship to the statutorily qualifying person beyond the ordinary consequences of separation or relocation, assessed in the totality of relevant circumstances. The applicable statutory provision determines whose hardship matters. See USCIS Policy Manual, Vol. 9, Part B, `https://www.uscis.gov/policy-manual/volume-9-part-b`.

[2] *Well-founded fear of persecution* is an asylum-law standard requiring a reasonable possibility of future persecution on account of a protected ground, evaluated using the applicant's circumstances and supporting evidence; it does not require a greater-than-50% probability of persecution. See 8 CFR § 208.13, `https://www.ecfr.gov/current/title-8/section-208.13`.

Aicumene:2026.040v10 [cs.AI] October 2026

for interpretive content, but the acceptance gate for formally decidable claims is never a model. Section 12 gives the reason in one inequality: a critic that reads the proposer's sources cannot detect the errors that originate there.

Compound AI systems and runtime enforcement. State-of-the-art results increasingly come from systems of interacting components rather than from a single model [28]; the architecture here is such a system with one addition, an explicit allocation of *authority* among components. The guarantees of Section 11 are specializations of standard results: guarded forward invariance is the discrete-time form of the invariance guaranteed by control barrier functions [6] and of shielding in reinforcement learning [5]; sound abstraction comes from symbolic verification of hybrid systems [27]; and checking executions against a specification at run time is the subject of runtime verification [22]. Our contribution is their application to a legal agent whose safety-relevant state includes matter identifiers, permissions and approval records.

Economics of verification and model behaviour. In a companion paper on verification games we prove the common-source floor in general and show that, because a public verdict is non-excludable, independent verification is systematically under-purchased on an explicit band of prices [17], and that the same wedge recurs in sequential reliance, in verification markets and across provenance structures [18]; Section 12 applies the floor to the compiler. In cross-family experiments with language-model agents we find that such agents reach the equilibrium of a game exactly when it reduces to a myopic best response over explicit payoffs [16]; this is the empirical reason for drawing the Critic from a different model family than the Proposer (Section 8).

Professional responsibility. The American Bar Association's Formal Opinion 512 holds that the existing duties of competence, confidentiality, communication, supervision, candor and reasonable fees apply in full to a lawyer's use of generative AI tools [1]. The architecture is designed to make those duties dischargeable: every machine-verified conclusion is cited and reconstructable, confidential matter data never leave a scoped data plane, and every interpretive question reaches a human.

Aicumene:2026.040v10 [cs.AI] October 2026

3 Approach: one deterministic arbiter

The language model proposes a candidate analysis; a deterministic compiler checks the encoded predicates and controls release of conclusions. A bright-line *rule* is independently recomputed; an open-textured *judgment* is referred to a qualified human rather than automatically asserted. Rather than asking a second LLM to police the first and treating agreement as reliability, we ask what parts of the legal workflow admit *independent deterministic disposal*. The principle is concise: *the model proposes; the compiler disposes.* Generation remains useful, but correctness of formally decidable claims is delegated to a non-generative oracle.

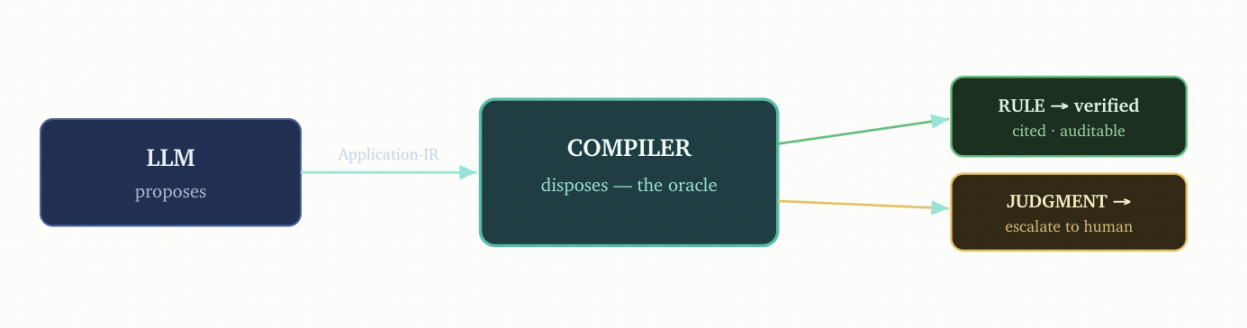

Figure 1: Approach—the compiler separates mechanically verifiable rules from judgments requiring human review.

4 Architecture: three axes around the arbiter

The first axis is the *verifiability compiler*; the second is *security and isolation*; the third is a *capability-scoped data plane* joining law, agency guidance, executable rules, and per-matter evidence. The verifier remains outside the model's action runtime. Earlier engineering work on protected remote robotic systems [14] provides relevant motivation for separating privileged verification from potentially fallible action runtimes; it does not establish the security of this legal prototype. The surrounding question of how intelligence and knowledge-production systems are organized in a changing sociotechnical environment is examined at a broader level in the collective report *The Entangled World* [2, 3] and, for AI specifically, in our companion paper on verifiable intelligence ecologies [15]; the present paper addresses a narrower, implementable verification boundary.

The axes are complementary. The compiler without source governance cannot know whether it formalized the right source. Isolation without workflow control still permits premature completion. A capability-scoped data plane makes the verifier reproducible and constrains evidence exposure; it also determines how many of the proposer's sources the verifier shares, which Sections 12 and 13 show to be the quantity that governs the verifier's value.

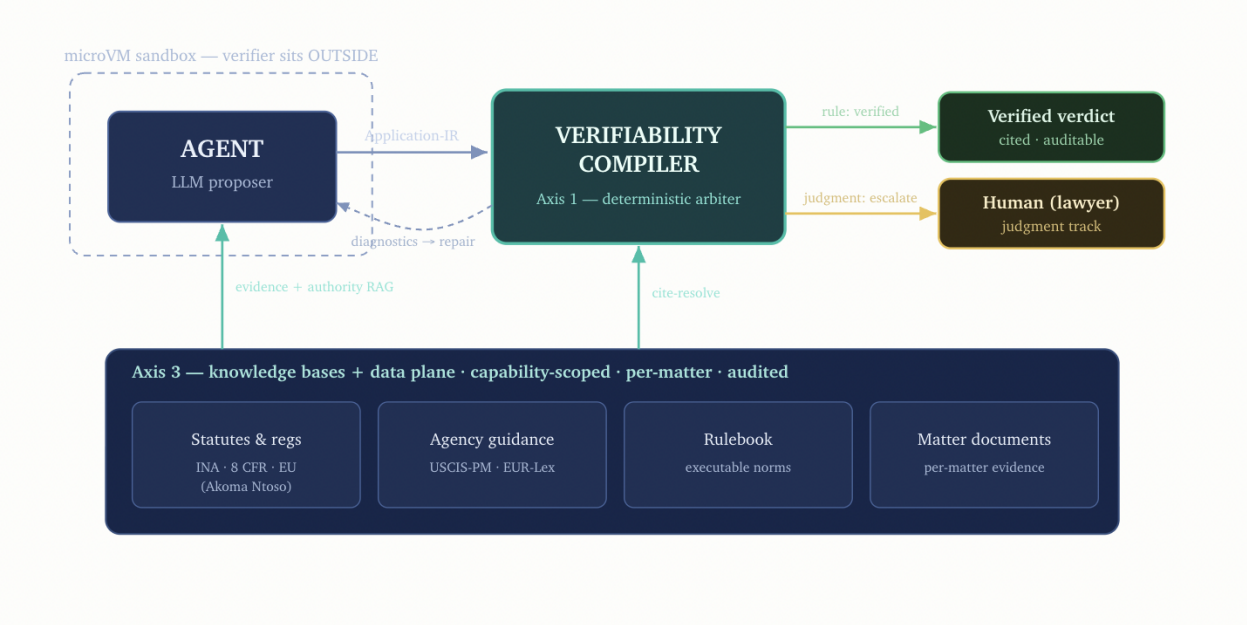

Figure 2: Architecture—three axes meet at an independent deterministic arbiter.

5 Axis 1: the verifiability compiler and the Application-IR

The compiler receives a typed `Application-IR` rather than a raw conversational answer. We write it as

$$\mathcal{I} = (\mathsf{F}, \mathsf{R}, \mathsf{E}, \mathsf{C}), \tag{1}$$

where F is a set of asserted facts, R a set of cited legal rules, E provenance links from facts to evidence and from rules to authority sources, and C the candidate verdict. The rules in R are drawn from an executable rulebook in the sense of [25]: each rule carries the identifier and version of the Akoma Ntoso text it formalizes [4]. The compiler's successive gates parse, resolve citations, establish jurisdiction, recompute thresholds, check whether sources are in force, and classify a proposition as a determinate rule or an evaluative judgment. Each gate emits a diagnostic

$$D_j = (c_j, s_j, \ell_j, h_j) : \quad \texttt{Diagnostic\{code, severity, location, hint\}}, \tag{2}$$

Aicumene:2026.040v10 [cs.AI] October 2026

where c_j is a code, s_j a severity class, ℓ_j the implicated location or claim, and h_j a hint for repair or escalation. Severity is not a user-interface convenience: an `error` blocks or repairs a deterministically incorrect rule claim; a `warning` triggers escalation for legal judgment. The compiler's output on a decidable claim is

$$\mathrm{Ver}(\mathcal{I}) \in \{\mathsf{Verified}, \mathsf{Rejected}, \mathsf{Escalated}, \mathsf{InsufficientEvidence}\}, \tag{3}$$

and "Verified" is intentionally narrow: the specified formalized condition is satisfied under authenticated premises and the selected source version. It does *not* mean that the matter as a whole has been adjudicated. This architecture checks conformity with an approved rulebook; it does *not* prove that the rulebook itself is legally correct.

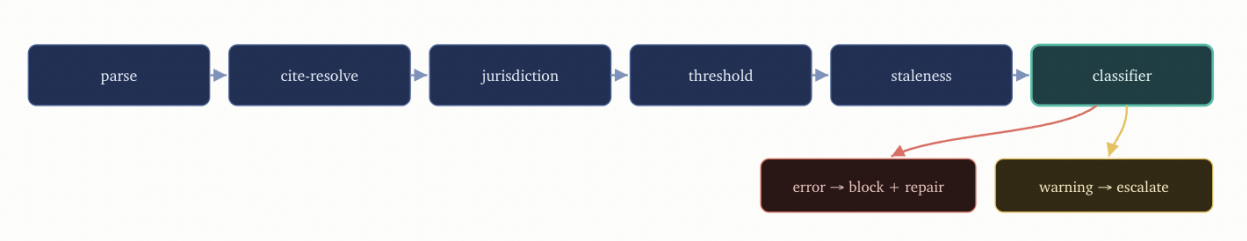

Figure 3: Compiler gates—each check produces a location-aware diagnostic and a controlled next step.

6 The rule/judgment firewall

For a determinate numerical test such as "at least 913 days present?", the compiler recomputes the result from validated facts and an approved rule. For an evaluative question such as "extreme hardship?", it declines to issue an autonomous substantive conclusion and escalates. The numerical threshold is an *illustrative encoded test*, not a jurisdiction-independent legal requirement. We model the routing of a question q by

$$\Phi(q) = \begin{cases} \mathrm{Ver}(q), & q \in \mathcal{Q}_{\text{decidable}}, \\ \mathrm{Rev}(q), & q \in \mathcal{Q}_{\text{judgment}}, \\ \mathrm{Abs}(q), & q \in \mathcal{Q}_{\text{unsupported}}, \end{cases} \tag{4}$$

where Rev is authorized human review and Abs requests additional evidence or abstains. The category assignment itself must be governed: converting a judgment question into a structured field does not make it decidable. The firewall is the core jurisprudential boundary of the architecture. A valid legal system may support a judgment question with evidence and structured reasoning, but it must not auto-assert it as if it were a bright-line arithmetic condition.

7 Knowledge bases and two retrieval paths

Four bases are logically segregated: *statutes and regulations* (e.g., U.S. INA and 8 CFR; relevant EU instruments), *agency guidance* (e.g., USCIS Policy Manual and EUR-Lex), an *executable rulebook*, and *per-matter client documents*. Authority RAG resolves the norm and its effective date; Evidence RAG establishes facts from the authorized matter. Retrieval uses capability-scoped access, logs requests and returned identifiers, and preserves provenance: every material fact maps to evidence; every rule maps to its source.

Legal grounding is *dual*: one must ground both in authority and in matter evidence, and retrieving a source is not the same as authorizing a conclusion. A statute may be relevant yet not in force; guidance may be persuasive yet non-binding; matter evidence may support a fact but not establish a legal classification. The system therefore preserves provenance at every step: each fact points to evidence, each executable rule to an authoritative source identifier and version, and each conclusion to the diagnostics that permitted it. RAG

Aicumene:2026.040v10 [cs.AI] October 2026

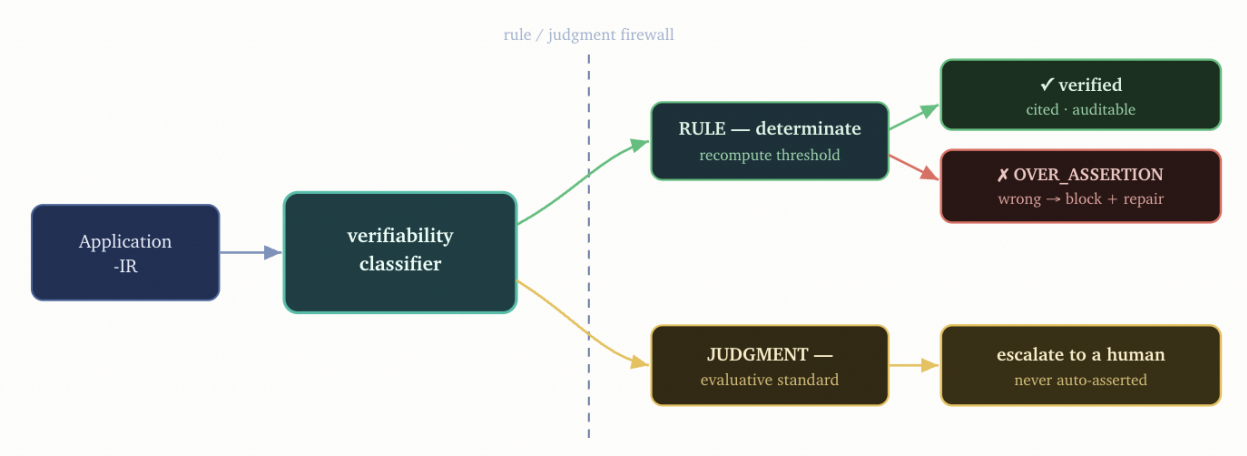

Figure 4: Rule/judgment firewall—different legal propositions receive different kinds of treatment.

is thus a *governed data plane*, not a convenience layer: the proposer has no unconstrained filesystem, database or network access, operates only over the current matter and the approved authority bases, and every retrieval is logged. Matter isolation is a first-class part of correctness: a conclusion derived from evidence that should never have been visible to the agent is institutionally invalid even if linguistically persuasive.

Source snapshots. For the guarantees of Section 12 one more notion is needed. A *source snapshot* is the versioned set of authorities and rule encodings that a component reads. The proposer's snapshot and the compiler's snapshot may coincide, overlap, or be disjoint; the share of the proposer's errors that originate inside the compiler's snapshot is the quantity c of Section 12, and the data plane is where it is set.

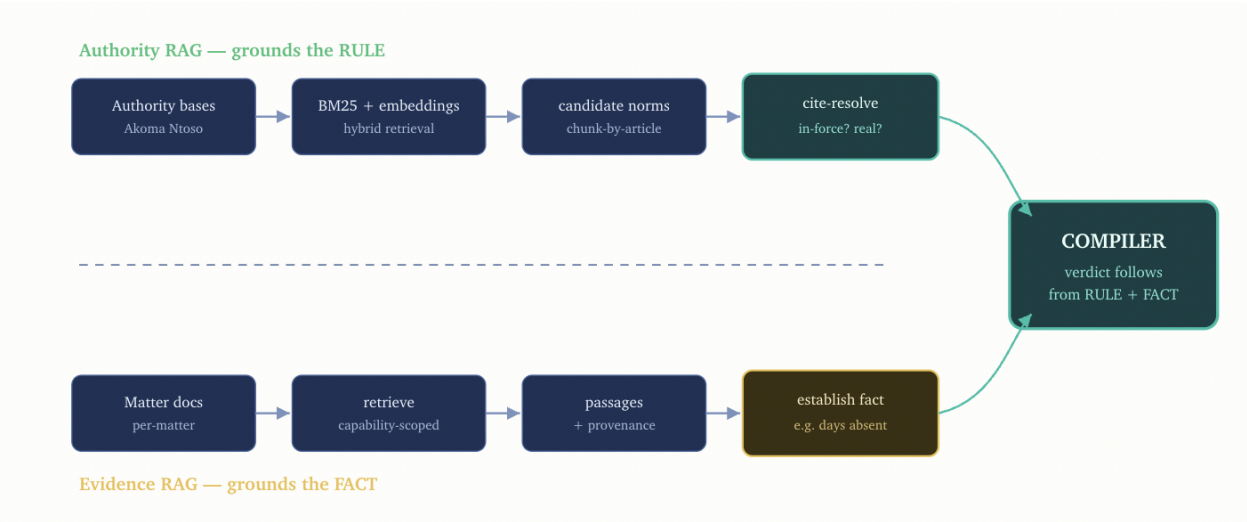

Figure 5: The two retrieval paths—authorities establish the applicable rule; documents support case-specific facts.

Aicumene:2026.040v10 [cs.AI] October 2026

8 Three agent roles: a grounded dialectic

The **Proposer** (a Qwen-based agent in the described prototype) generates a structured candidate. The **Critic**, preferably from a different model family, attempts to refute assertions and expose gaps. The **Compiler** resolves mechanically verifiable issues against encoded sources rather than taking a vote among LLMs. Criticism can improve proposals but does not become legal authority.

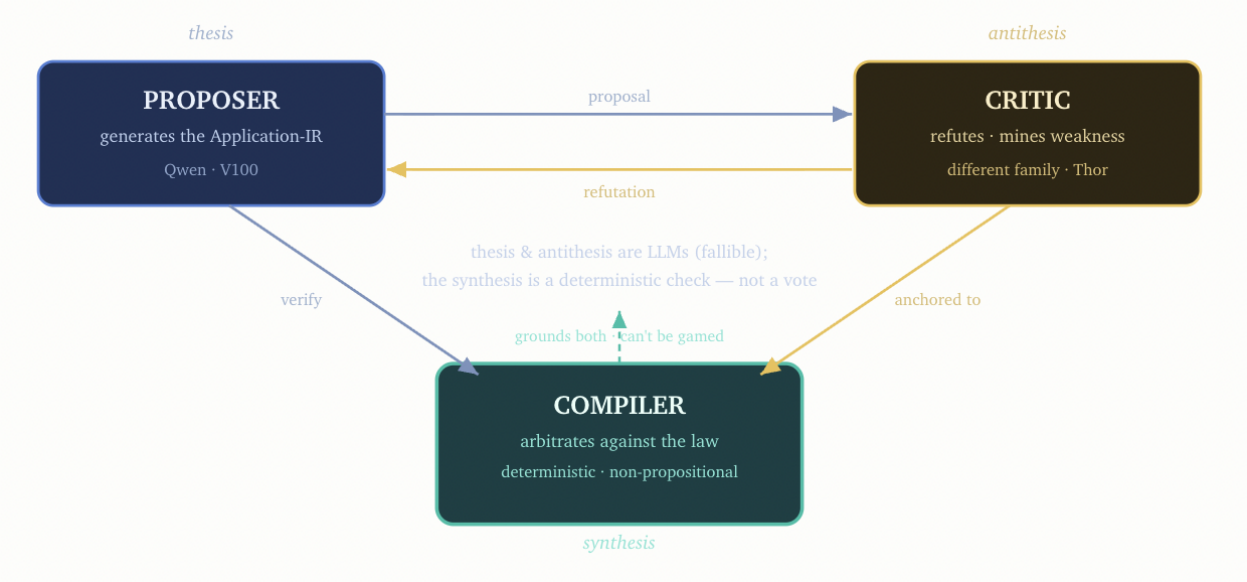

Figure 6: Proposer–Critic–Compiler. The proposer generates candidate Application-IR; the critic challenges its claims; the independent compiler verifies formally decidable conditions against approved rules and validated facts, or escalates questions requiring human legal judgment. The synthesis is a deterministic check, not a vote between language models.

Independent verification boundary. The Proposer and Critic can expose gaps and improve candidate analyses, but agreement between generative models is not an independent legal authority. The compiler instead recomputes formally decidable predicates using an approved executable rulebook and validated, provenance-linked facts. It emits traceable diagnostics, blocks unsupported mechanically checkable conclusions, and escalates questions requiring contextual legal evaluation to qualified counsel. This is a limited verification boundary, not an oracle of legal truth: correctness remains conditional on current and accurate legal sources, correctly encoded rules, and authentic, sufficiently complete evidence. The distinction between truth and evidence-mediated access to it is discussed more broadly in the April Theses [7].

Why the critic should come from another family. Two language models of one family share training data, defaults and dispositions; in our cross-family experiments the strategic behaviour of agents differed systematically by family and the differences were invisible in self-play [16]. A critic from another family is therefore more likely to disagree where disagreement is informative. This is a statement about the critic's usefulness, not about verification: Section 12 shows that even a critic from another family cannot detect an error that originates in a source both models read.

Aicumene:2026.040v10 [cs.AI] October 2026

Component	Function	Authority limit
Proposer	Candidate facts, norms and analysis	Cannot certify itself
Critic	Detect unsupported assumptions	Cannot replace verification
Retriever	Locate candidate authorities and evidence	Retrieval does not prove applicability
Compiler	Evaluate formal conditions	No discretionary legal judgment
Workflow kernel	Preserve state, deadlines and gates	Executes only authorized transitions
Human reviewer	Resolve interpretive issues	Decision recorded with provenance

9 A harness that improves without gaming the evaluator

The agent can propose revisions to its prompts, tools, or harness after inspecting failures; a held-out compiler-anchored evaluation gate accepts only validated improvements. Acceptance is gated by held-out evaluation anchored to the deterministic verifier and workflow metrics, not by self-reported improvement; otherwise a system can optimize for looking correct to itself. The published note reports a critic trust measure increasing from 0.47 to 0.65 over training epochs. This is a *reported internal prototype indicator*, not a general accuracy result or a cross-system benchmark. Its validity depends on the coverage and correctness of the fixed compiler anchor.

10 Long-horizon matters: stopping is verified

Complex matters require gap finding, additional retrieval, and repeated testing of unmet preconditions. Let a matter require rule prongs $\theta_1, \ldots, \theta_n$ with dispositions in $\{\mathsf{verified}, \mathsf{failed}, \mathsf{escalated}, \mathsf{unresolved}\}$. The workflow may close the *analysis stage* only when every θ_i has an explicit disposition other than $\mathsf{unresolved}$, every judgment prong has been referred for review, and all mandatory approvals are recorded. Technical completion is a verified state of the workflow kernel, not the model's confidence. Closure does not imply government approval or ultimate legal correctness; it guards against declaring success after an incomplete chain of analysis.

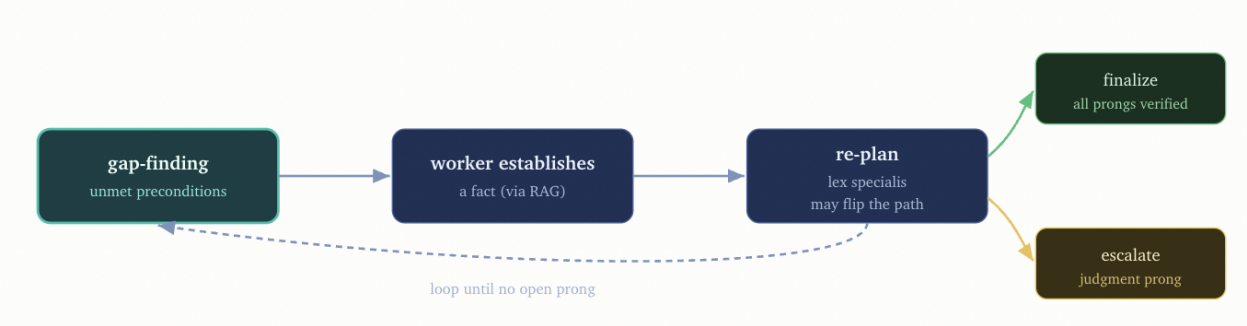

Figure 7: Long-horizon stopping—the kernel, not the model, determines whether a matter is technically complete.

11 Axis 2: isolation and security, and its formal basis

Related prior work on cybersecurity for remotely operated robotic systems [14] motivates a defense-in-depth perspective; a personalized surgical-simulation workflow [24] likewise illustrates the engineering need to separate model outputs, inputs, and authorized actions, without constituting validation of the present legal security design.

The proposed perimeter combines LAN/mTLS, default-deny egress, secrets management, per-matter privileges, and hash-chained audit. The intended microVM progression is gVisor to Firecracker. The action

Aicumene:2026.040v10 [cs.AI] October 2026

runtime is sandboxed; the compiler is outside the isolation boundary. Full production microVM enforcement is described as a *planned stage*, distinct from existing per-matter segregation in code.

What isolation can guarantee. Represent the tool-using agent by a controlled transition system $x_{t+1} = T(x_t, u_t, w_t)$ with state space X (matter identifiers, permissions, retrieved evidence, workflow stage, external-system state), admissible actions U, environment inputs W, initial states X_0 and a set $B \subseteq X$ of forbidden states: for instance, states in which one client's documents have been disclosed to another matter, an unapproved filing has been submitted, or the rule library has been modified without authorized review. Safety is the invariance statement $x_t \notin B$ for all t. Let $\Omega \subseteq X \setminus B$ be a certified invariant set and $\Gamma(x) \subseteq U$ the actions a trusted guard allows at x.

Proposition 1 (Guarded forward invariance)**.** *Suppose $X_0 \subseteq \Omega$ and, for every $x \in \Omega$, every $u \in \Gamma(x)$ and every admissible $w \in W$, $T(x, u, w) \in \Omega$. If all externally consequential actions are mediated by an enforcement component that permits only $u \in \Gamma(x)$, then every reachable state lies in Ω and no forbidden state is reached.*

Proof. $x_0 \in \Omega$; if $x_t \in \Omega$ the enforced action lies in $\Gamma(x_t)$, so $x_{t+1} \in \Omega$ for every admissible w_t; induction completes the argument, and $\Omega \cap B = \varnothing$. □

This is the discrete-time form of control-barrier invariance [6] and the mechanism of a shield [5]; Γ is the shield. Its strength is also its limitation: a guard cannot enforce conditions on unobserved or unauthenticated facts. The enforcement point must therefore receive authoritative matter identifiers, permission records, tool parameters and approval state, not a model-generated statement that authorization exists.

Remark 1 (Observation indistinguishability). If the monitor sees only telemetry $y = h(x)$ and there are reachable states $x_s \notin B$ and $x_b \in B$ with $h(x_s) = h(x_b)$, then no classifier of the current observation is correct on both. A high-dimensional monitor can still omit one decisive permission bit; adding reliable history, sensors or enforcement gates refines h and can resolve the ambiguity. Every tool introduced into the harness must come with a transition specification, or a conservative over-approximation in the sense of a simulation relation [27], that preserves the invariant; unmodeled effects defeat the guarantee, and checking executions against such specifications at run time is runtime verification [22]. The *verification coverage gap* G_V, the probability on a reference distribution of safety-relevant transitions that a transition lacks a validated enforcement rule, is the measurable quantity to report.

12 What the compiler guarantees: conditional soundness and the common-source floor

For a task distribution P over questions, let $\mathcal{Q}_V$ contain the questions with valid independent verification and define the *verification coverage* $\kappa_V = \Pr_{q \sim P}(q \in \mathcal{Q}_V)$. Coverage differs from model accuracy and from the fraction of outputs labelled "verified." A deterministic checker produces consistent yet incorrect outcomes when the rule, the evidence or the temporal applicability is wrong. Soundness is therefore *conditional*: on authenticated premises, correctly encoded current authority, correct source dating and correct implementation.

Proposition 2 (Conditional soundness)**.** *Fix a matter, its jurisdiction and effective date. If every rule in* R *correctly formalizes the authority it cites as in force on that date, every fact in* F *is authenticated, and the compiler implements its gates correctly, then every claim it marks* Verified *is true, and every claim it marks* Rejected *is false, with respect to the formalized rule. Conversely, a* Verified *claim that is false identifies a violated hypothesis: a mis-encoded or superseded rule, an unauthenticated fact, or an implementation defect, each of which the diagnostic's location field localizes.*

Proof. Under the hypotheses the gates evaluate a correct predicate on correct inputs; the contrapositive is immediate. □

The hypotheses are not free, and the first of them is where the proposer and the compiler can fail *together*. Let the proposer be wrong on a claim with probability p, and let a share c of its errors originate in a source that the compiler also reads: a stale edition of a regulation, a mis-encoded threshold, an agency page that

Aicumene:2026.040v10 [cs.AI] October 2026

both components retrieved. On that event the compiler's inputs are exactly what they would be if the claim were correct.

Proposition 3 (Common-source floor)**.** *Let* FA *be the probability that a wrong claim is accepted and* FR *the probability that a correct claim is rejected, under any decision rule that uses only the compiler's reads and any amount of further checking against the same sources. Then*

$$\mathrm{FA} \;\geq\; p\,c\Big(1-\frac{\mathrm{FR}}{1-p}\Big), \tag{5}$$

and in particular $\mathrm{FA} \geq pc$ *for every rule that accepts correct claims. The bound does not depend on the compiler's own error rate, on the number of checks, or on how many language models agree.*

Proof. On the shared-source event the joint law of everything the compiler reads is the same as on a correct claim, so any rule accepts with the same probability $a = 1 - \mathrm{FR}/(1-p)$ on both; hence $\mathrm{FA} \geq pc\,a$. The general statement, its extension to sources that copy one another, and its consequences for the purchase of independent verification are proved in [17]. □

Corollary 1 (Source governance is first-order)**.** *If the compiler's rule-encoding error is* p_{rule}*, independent of the proposer, and claims are accepted iff the compiler accepts them, then on a population in which half the claims are wrong*

$$\mathrm{FA}_{\mathrm{guarded}}(c, p_{\mathrm{rule}}) = \tfrac{1}{2}\big[p\,c + (1-p\,c)\,p_{\mathrm{rule}}\big], \qquad \mathrm{FA}_{\mathrm{unguarded}} = \tfrac{1}{2}\,p. \tag{6}$$

The term $p\,c$ *is the floor of Proposition 3; it is lowered only by reducing* c*, that is, by giving the compiler a source snapshot the proposer does not read. At* $c = 1$ *the compiler adds nothing and, with* $p_{\mathrm{rule}} > 0$*, subtracts.*

The practical reading is that the data plane of Section 7, which sets c, is a first-order control, and that a compiler whose rulebook is derived from the same retrieval that feeds the proposer verifies very little. The same inequality is why Proposer–Critic agreement is not verification: a critic from another model family that reads the same authorities inherits the same floor.

13 A control-layer experiment with a fallible verifier

13.1 Why an oracle test proves nothing

A legal workflow differs from a classification task: the output may be an unauthorized *action*, an unsupported citation or a premature closure, not merely a wrong label. An earlier draft of this work contained a "smoke test" in which a guard read the planted ground truth of every fixture and, unsurprisingly, produced a false-accept rate of zero. That result was tautological and is withdrawn: the guard was an oracle. The experiment below makes the condition of Proposition 2 the independent variable.

13.2 Design

Fixtures carry a planted Boolean ground truth and a proposal produced by a noisy process that flips the truth with probability $p = 0.18$; no language model is involved. Five fixture classes are generated (arithmetic predicates, citation validity, judgment questions, completion state, cross-matter authorization), 240 per class per seed, over 30 seeds. Judgment fixtures are always escalated and never auto-accepted; the remaining four classes are decidable and are the object of measurement. The verifier is fallible in the two ways of Section 12: a share c of the proposal's errors are shared-source errors that it inherits, and it makes its own rule-encoding errors with probability p_{rule}, independently of the proposal. The guarded mode accepts the verifier's verdict; the unguarded mode accepts the proposal as it is. The grid is $c \in \{0, 0.1, 0.3, 0.5, 0.75, 1\}$ and $p_{\mathrm{rule}} \in \{0, 0.01, 0.05\}$; the script `experiments/legal_benchmark.py` and the seed-level results accompany the package, and Table 1 is generated from its summary file.

Aicumene:2026.040v10 [cs.AI] October 2026

13.3 Results

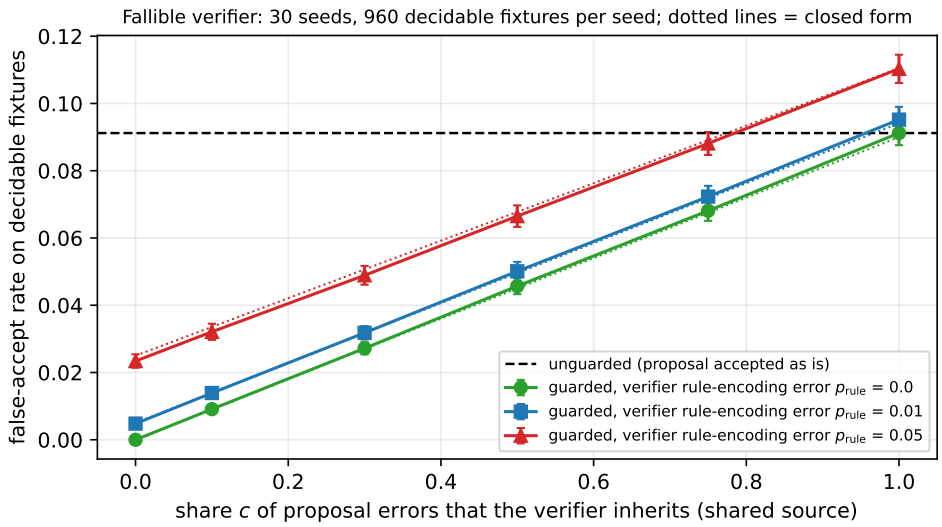

Figure 8: False-accept rate on decidable fixtures as a function of the shared-source share c, for three verifier rule-encoding error rates. Bars are two standard errors over 30 seeds; dotted lines are the closed form (6); the dashed line is the unguarded rate.

Mode	$(c,\ p_{\text{rule}})$	false accept	±SE	closed form (6)
unguarded	—	0.091	0.002	0.090
guarded (oracle corner)	(0, 0)	0.000	0.000	0.000
guarded	(0, 0.01)	0.005	0.000	0.005
guarded	(0, 0.05)	0.023	0.001	0.025
guarded	(0.3, 0.01)	0.032	0.001	0.032
guarded	(0.5, 0)	0.046	0.001	0.045
guarded	(1, 0)	0.091	0.002	0.090
guarded	(1, 0.05)	0.110	0.002	0.111

Table 1: Selected cells of the fallible-verifier experiment (30 seeds, 960 decidable fixtures per seed). False-reject rates are symmetric and omitted.

Figure 8 and Table 1 show three things. The simulation follows the closed form to within sampling error, so the experiment is a check of the accounting, not a source of surprise. The value of a verifier is governed by c far more than by p_{rule}: a verifier that inherits half the proposer's sources with perfect rule encoding is worse than a verifier with independent sources and a 5% encoding error rate (0.046 against 0.023). And at $c = 1$ the verifier adds nothing (0.091 against 0.091 unguarded) and, with its own errors, subtracts (0.110). This is a result about the control layer in a synthetic environment; it says nothing about the accuracy of any legal reasoning system.

14 Prototype demonstrations and rollout

The published project note reports end-to-end demonstrations on a Qwen model running on an NVIDIA V100. In an *EU long-term-residence* scenario, three client documents were ingested, a duration of "six years three months" was represented as 6.25 years, and the prototype reached a compiler-approved demo label `GRANTABLE`. In an *asylum* scenario, a mechanically checkable filing element was evaluated while the substantive

Aicumene:2026.040v10 [cs.AI] October 2026

"well-founded fear" question was escalated. These are demonstrations of workflow behavior, *not verified legal eligibility opinions.*

The development plan distinguishes a completed prototype core (P0) from planned authority retrieval (P1), microVM hardening (P2), human escalation interface (P3), matter-management integration (P4), and EU-first rulebook authoring (P5).

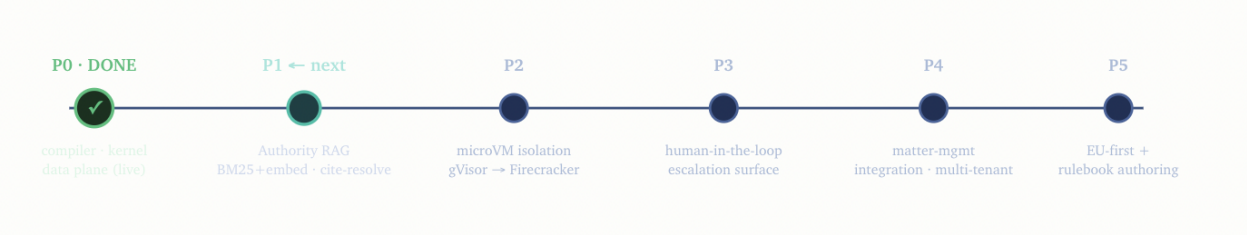

Figure 9: Architectural rollout—the note separates the currently described prototype from intended future components.

15 Evaluation protocol for a genuine legal benchmark

The experiment of Section 13 fixes the upstream error process and varies the verifier; it involves no language model and no legal matter. The step from it to an empirical claim about legal AI requires the following protocol, stated so that it can be pre-registered, in the manner of [23].

Materials. Cases are version-controlled matter bundles with authoritative source snapshots, factual evidence, jurisdiction, and a list of determinate rule prongs and judgment prongs. Qualified legal reviewers produce a source-linked reference annotation and record disagreements; adjudication is kept separate from model-generated recommendations. Splits are by matter, so that duplicate filings, templates or the same client never cross partitions. Source updates, missing evidence, jurisdiction switches and unauthorized tool requests appear as stress conditions.

Factorial design. Four systems run on identical held-out bundles with matched tool permissions, inference budgets and retrieval corpora: (A) a generalist with ordinary retrieval; (B) specialized but unverified agents; (C) a hybrid with shared reasoning and bounded specialists; (D) the same hybrid with the verifiability compiler, the rule/judgment firewall and human escalation. The contrast B versus A isolates specialization; D versus C isolates independent verification; a better guard is not evidence that a smaller model learned the task better. The compiler's c is manipulated by giving its rulebook either an independent source snapshot or the same retrieval the proposer uses, so that (6) can be tested on real outputs; under the rule "accept iff the compiler accepts" with a common own error q, the difference in false-accept rates between the two arms is $p\,c\,(1-2q)$, which identifies c up to the own error.

Outcomes. (a) correct completion of decidable prongs; (b) false acceptance of incorrect rule applications; (c) citation existence and temporal validity; (d) premature-closure rate; (e) judgment treated as determinate; (f) cross-matter exposure under adversarial prompting; (g) compute and human-review cost per completed, independently checked matter. Differences are reported with confidence intervals over matters, paired by matter where the same bundle is seen by all systems.

Hypothesis 1 (Specialization efficiency)**.** *On narrowly specified tasks, specialist or deterministic components produce competitive or superior verified performance per unit of compute relative to generalist-only execution (B versus A).*

Hypothesis 2 (Independent verification)**.** *A sound external compiler with an independent source snapshot reduces incorrect acceptance of decidable claims compared with generative self-evaluation (D versus C), by an amount that shrinks toward zero as the compiler's sources are made to coincide with the proposer's.*

Aicumene:2026.040v10 [cs.AI] October 2026

Hypothesis 3 (Stateful completion)**.** *Explicit completion gates reduce the rate of premature finalization of long-horizon legal workflows.*

Hypothesis 4 (Controlled delegation)**.** *Capability-scoped orchestration reduces unauthorized actions and cross-matter disclosures compared with unrestricted tool access.*

Hypothesis 5 (Hybrid Pareto frontier)**.** *For mixed legal tasks, a combination of generalist interpretation and specialized verification improves the performance–risk–cost Pareto frontier relative to pure generalist and unverified multi-agent baselines.*

No hypothesis is presented as experimentally proven. No new legal-case benchmark scores are reported in this paper.

16 Scope, limitations, and professional responsibility

All legal thresholds, effective dates, and citations must be independently verified by qualified counsel before authoritative use. The compiler only decides what can be expressed as reliable executable predicates over validated facts; it cannot autonomously resolve open-textured legal standards or ensure that an encoded rule is legally correct, and Proposition 3 shows that no amount of checking against shared sources can. Claims about self-improvement are epoch-local. The control-layer experiment is synthetic: its proposal errors are coin flips, its verifier is a two-parameter abstraction, and its closed form holds because the fixtures are independent; it establishes the accounting, not the behaviour of any deployed system. Human approval can fail if reviewers see incomplete evidence or are prompted to rubber-stamp outputs; the architecture therefore requires independent source governance, meaningful review interfaces and external evaluation.

The system is presented for *research and legal decision support, not autonomous legal advice.* It is designed to be used by lawyers within their existing duties [1]: competence requires understanding what the compiler does and does not check; confidentiality is served by the matter-scoped data plane and must still be secured contractually with any model provider; communication and candor require that machine-verified and human-judged conclusions be labelled as such to clients and tribunals; supervision applies to the system as to any non-lawyer assistant. Whether a given deployment constitutes the unauthorized practice of law is a question of the jurisdiction's rules on who may render advice, not of the architecture, and must be answered before any deployment that reaches members of the public.

In one line. *The model proposes; the compiler disposes. Every machine-verified verdict should be checkable, cited, and reconstructable.*

17 Code and data availability

The source code and implementation of the legal verifiability compiler, including the rule/judgment firewall, are available in the Aicumene GitHub repository `github.com/aicumene/verifiable-law`, which contains the implementation and supporting materials for the Verifiable AI for Law research project. The fallible-verifier experiment of Section 13 is the script `experiments/legal_benchmark.py`, distributed with this paper together with its seed-level results; it needs only NumPy and Matplotlib, and the table and the numbers quoted in Section 13 are written by the script. Only synthetic fixtures generated algorithmically were used; no client matter, privileged file or lawyer-reviewed benchmark.

18 Conclusion

Verifiable AI for Law presents an architecture for separating generative legal analysis from independently executable checks. The Proposer drafts a structured candidate, the Critic challenges its premises and supporting evidence, and the compiler determines whether formally decidable conditions satisfy approved rules and validated facts. This arrangement does not treat agreement between language models as legal verification, and the paper says why in one inequality: a verifier that reads the proposer's sources cannot detect the errors that originate there, so the value of verification is governed by the independence of its sources, not by the number or the quality of the checks.

Aicumene:2026.040v10 [cs.AI] October 2026

The paper makes four connected contributions. First, a typed *Application-IR* makes assertions, legal authorities, factual predicates, and proposed outcomes explicit. Second, an independent *verifiability compiler* checks encoded conditions and produces reproducible, source-linked diagnostics, with a stated conditional-soundness guarantee. Third, the *rule/judgment firewall* prevents evaluative standards, including "extreme hardship" and "well-founded fear," from being mistaken for mechanically decidable rules; these questions are routed to qualified human reviewers. Fourth, the common-source floor and the experiment that confirms it make source governance a measurable first-order control, and the evaluation protocol makes the architecture's claims falsifiable on real matters. Together with segregated authority and evidence retrieval and capability-scoped execution, these components support auditable legal workflows and help prevent premature completion.

The epistemic motivation is consonant with the discussion of evidence-based access to truth in the April Theses [7]. At a different technical level, VeNet [8], together with the associated annotated MRA dataset [10] and HessNet preprint [11], provides examples of explicitly structured computation in a compact neural model; this is an architectural analogy, not a validation of legal reasoning. The deterministic layer of the compiler belongs to the same line as counter-augmented automata for real-time regular-expression matching without state explosion [12, 9], and the monitoring side of the firewall can use the topology-driven classification of time series [13] as a structural readout of a running system. The broader sociotechnical context is examined by Ageev, Gromov, the Pereslegins, and colleagues [2, 3].

The compiler is not a source of legal authority: its results depend on correct and current rule encoding, reliable retrieval, authentic facts, and sufficient evidence. Open-textured legal interpretation and consequential decisions still require qualified human review. Prototype demonstrations illustrate workflow behavior rather than independently established legal eligibility or generalizable accuracy. Further work should validate rulebooks with counsel, run the protocol of Section 15 across jurisdictions, harden isolation and provenance controls, and evaluate human escalation procedures. The goal is a shift from unqualified AI-generated answers toward *verifiable legal computation within explicit limits. The model proposes; the compiler disposes.*

Aicumene:2026.040v10 [cs.AI] October 2026

Declarations

Author contributions. Both authors contributed to the architecture, the formal statements, the experiment and the writing. **Competing interests.** The authors are associated with research and commercial development in legal AI, including Aicumene; no product evaluation is reported. **Funding.** No external funding. **Figures.** Figures 1–8 are schematic drawings produced for this paper; Figure 8 is generated by the experiment script. No generative-AI imagery is used.

References

[1] American Bar Association, Standing Committee on Ethics and Professional Responsibility (2024). Formal Opinion 512: Generative Artificial Intelligence Tools. 29 July 2024. https://www.americanbar.org/content/dam/aba/administrative/professional_responsibility/ethics-opinions/aba-formal-opinion-512.pdf.

[2] Ageev, A. I., Gromov, V. A., Lukovnikova, N. M., Pereslegin, S. B., Pereslegina, E. B., Shilov, S. Yu., et al. (2026). *Spletennyi mir: koshka perevernulas' [The Entangled World: The Cat Has Turned Over]. Report to the Club of Rome.* St. Petersburg: Polaks. ISBN 978-5-905898-66-2. https://publications.hse.ru/books/1195670107.

[3] Ageev, A. I., Gromov, V. A., Lukovnikova, N. M., Pereslegin, S. B., Pereslegina, E. B., Shilov, S. Yu., and Yutanov, N. Yu. (forthcoming 2027). *The Woven World: Complexity Crisis and Global Futures.* World-Systems Evolution and Global Futures. Cham: Springer. ISBN 978-3-032-36259-9. https://link.springer.com/book/9783032362599.

[4] OASIS (2018). *Akoma Ntoso Version 1.0. Part 1: XML Vocabulary* (eds. M. Palmirani, R. Sperberg, G. Vergottini, F. Vitali) and *Part 2: Specifications* (eds. F. Vitali, M. Palmirani, R. Sperberg, V. Parisse). OASIS Standard, 29 August 2018. https://docs.oasis-open.org/legaldocml/akn-core/v1.0/os/part1-vocabulary/akn-core-v1.0-os-part1-vocabulary.html.

[5] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S. and Topcu, U. (2018). Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence* 32(1). https://arxiv.org/abs/1708.08611. doi:10.1609/aaai.v32i1.11797.

[6] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K. and Tabuada, P. (2019). Control barrier functions: theory and applications. *2019 18th European Control Conference (ECC)*, 3420–3431. doi:10.23919/ECC.2019.8796030.

[7] Bernadotte, A. (2026). *The April Theses of the New Cognitive Paradigm: Measurability of Intelligence, Language, and Probabilistic Truth. Toward Structural Diagnostics of Consciousness, Strong AI, and Creativity.* Aicumene preprint, Aicumene:2026.046v1. doi:10.13140/RG.2.2.20527.50087.

[8] Bernadotte, A. (2026). VeNet: a lightweight neural network for efficient brain vessel segmentation in endovascular robotic surgery. *Scientific Reports*, **16**, 24566. doi:10.1038/s41598-026-54176-x.

[9] Bernadotte, A. (2026). Counter-augmented deterministic automata for real-time regular expression matching without state explosion. *2026 12th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, 471–475. doi:10.1109/ICARA69401.2026.11480280.

[10] Bernadotte, A., Elfimov, N., and Menshikov, I. (2026). Semi-automatic annotation of brain vessels in magnetic resonance angiography images. *Scientific Data*, **13**, 41. doi:10.1038/s41597-025-06354-1.

[11] Bernadotte, A., Elfimov, N., Shutov, M., and Menshikov, I. (2025). Hessian-based lightweight neural network HessNet for state-of-the-art brain vessel segmentation on a minimal training dataset. arXiv:2508.15660. doi:10.48550/arXiv.2508.15660.

[12] Bernadotte, A. (2023). Real-time regular expression matching. arXiv:2308.10208. doi:10.48550/arXiv.2308.10208.

[13] Bernadotte, A. (2026). Topology-driven classification of time series. bioRxiv 2026.04.25.720787. doi:10.64898/2026.04.25.720787.

[14] Bernadotte, A. (2023). Cyber security for surgical remote intelligent robotic systems. *2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, 65–69. doi:10.1109/ICARA56516.2023.10126050.

[15] Bernadotte, A. and Menshikov, I. (2026). *From Model Scaling to Verifiable Intelligence: An Ecological Architecture of Specialized AI Systems.* Aicumene preprint, Aicumene:2026.101.

[16] Bernadotte, A. and Menshikov, I. (2026). When do language models reach Nash? Heterogeneous cross-family LLM agents in classic economic games. SSRN working paper 7637178 (June 2026). `https://ssrn.com/abstract=7637178`, `https://research.aicumene.com/papers/llm-nash/`, doi:10.2139/ssrn.7637178.

[17] Bernadotte, A. and Menshikov, I. (2026). Multi-Agent Verification Games: Shared Sources, Crowding Out, and the Price of Independence. Part I. Aicumene Preprint 2026.047v1; SSRN working paper 7682932 (October 24, 2026). `https://ssrn.com/abstract=7682932`, doi:10.2139/ssrn.7682932. `https://research.aicumene.com/papers/game-theory-1/`

[18] Bernadotte, A. and Menshikov, I. (2026). Multi-Agent Verification Games II: Information Entrenchment and Competitive Underprovision of Independent Evidence. Part II. Aicumene Preprint 2026.048v1; SSRN working paper 7747018 (October 24, 2026). `https://ssrn.com/abstract=7747018`, doi:10.2139/ssrn.7747018. `https://research.aicumene.com/papers/game-theory-2/`.

[19] Dahl, M., Magesh, V., Suzgun, M. and Ho, D. E. (2024). Large legal fictions: profiling legal hallucinations in large language models. *Journal of Legal Analysis* 16(1), 64–93. doi:10.1093/jla/laae003.

[20] Genesereth, M. (2009). *Computational Law: The Cop in the Backseat.* Stanford Center for Legal Informatics (CodeX), Stanford University. `http://logic.stanford.edu/complaw/complaw.html`.

[21] Lawsky, S. B. (2017). A logic for statutes. *Florida Tax Review* 21(1), 60–80. doi:10.5744/ftr.2017.0002.

[22] Leucker, M. and Schallhart, C. (2009). A brief account of runtime verification. *Journal of Logic and Algebraic Programming* 78(5), 293–303. doi:10.1016/j.jlap.2008.08.004.

[23] Magesh, V., Surani, F., Dahl, M., Suzgun, M., Manning, C. D. and Ho, D. E. (2025). Hallucination-free? Assessing the reliability of leading AI legal research tools. *Journal of Empirical Legal Studies* 22(2), 216–242. doi:10.1111/jels.12413.

[24] Menshikov, I., Orlov, K., Berestov, V., and Bernadotte, A. (2023). Preoperative personalized vascular 3D simulator of the intelligent robotic ecosystem LevshAI for endovascular neurosurgery. *2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, 6–10. doi:10.1109/ICARA56516.2023.10125981.

[25] Merigoux, D., Chataing, N. and Protzenko, J. (2021). Catala: a programming language for the law. *Proceedings of the ACM on Programming Languages* 5(ICFP), article 77, 1–29. doi:10.1145/3473582.

Aicumene:2026.040v10 [cs.AI] October 2026

[26] Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. and Cory, H. T. (1986). The British Nationality Act as a logic program. *Communications of the ACM* 29(5), 370–386. doi:10.1145/5689.5920.

[27] Tabuada, P. (2009). *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer. doi:10.1007/978-1-4419-0224-5.

[28] Zaharia, M., Khattab, O., Chen, L., Davis, J. Q., Miller, H., Potts, C., Zou, J., Carbin, M., Frankle, J., Rao, N. and Ghodsi, A. (2024). The shift from models to compound AI systems. Berkeley Artificial Intelligence Research blog, 18 February 2024. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

[29] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E. and Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems* 36. https://arxiv.org/abs/2306.05685. doi:10.52202/075280-2020. arXiv:2306.05685.

Citation

```
@misc{bernadotte2026verifiablelaw,
 title={Verifiable AI for Law: a legal verifiability
 compiler with a rule/judgment firewall},
 author={Bernadotte, Alexandra and Menshikov, Ivan},
 year={2026}, version={Aicumene:2026.040v10},
 howpublished={Aicumene Research preprint; SSRN},
 doi={10.13140/RG.2.2.18174.75841},
 url={https://research.aicumene.com/papers/verifiable-law/}
}
```

Aicumene:2026.040v10 [cs.AI] October 2026