Verifiable Legal Execution

Why legal AI needs authority, not intelligence:
a system model, its mechanisms and what the evidence shows

Whitepaper LL-WP-01
Version 1.0.0, 26 November 2026

Abstract

Legal work produces acts, not answers: filings, notices, approvals and findings whose consequences fall on people who did not write them. An act needs a warrant to be believed, which in law comes from an official source applicable on the relevant date, and a right to act, conferred by a mandate and exercised by a named person on exact content. Together these are authority, and intelligence is not authority: no gain in a model's competence turns its output into a source or its proposal into a mandate. Legal execution must make authority executable and refuse where it is missing.

We state a model of verifiable legal execution, with provisions as spans of admitted official texts, support over the decisive elements of a statement, release gates, approvals bound to tool, version, input hash, time window and named reviewer, and evidence chains. A gate multiplies the odds of a decisive error by its false-accept rate over its true-accept rate, so a verifier that accepts every decisive error cannot lower it; error and refusal compound.

Measured on LLRB-1, a pre-registered benchmark built by construction from eight EU acts, Lawie's grounding check accepted all 1,959 test statements carrying one legally decisive edit and all 698 cited to the wrong provision [MEASURED]. Calibrated lexical verifiers accepted 27.1–54.2% of decisive edits; a typed research prototype rejected every edit of the types it models and accepted 88.5% of the others; the reported confidence carried no information. Lawie's temporal components equate entry into force with applicability, which contradicts the schedules encoded from the official texts on 7–20% of provision-days. In 552 authorization scenarios the evaluated checks denied no legitimate request, refused every scenario of some attack classes and allowed every scenario of others. The audit-chain verifier detected every tested in-place edit, deletion, reordering, insertion and deleted prefix, and the dossier verifier rejected every tampered dossier.

Evidence boundary. Nothing here is deployed in production [BLOCKED] or has a dated production acceptance [ABSENT]; Civic Guard is quarantined [BLOCKED]; nothing, the benchmark labels included, is independently audited or expert-reviewed [ABSENT]; generated answers are not yet studied [PROPOSED]. All measurements are offline runs of component files at commit `ffa13d4bce` [MEASURED].

Prepared by Lawie Labs. *Status:* First edition. *Distribution:* Public. *Basis:* source repository `main` @ `ffa13d4bce` (2026-11-24); experiments of 26 November 2026; EUR-Lex texts retrieved 26 November 2026. *Audience:* Senior lawyers, legal technologists, security engineers, AI researchers, regulators and diligence teams.

Evidence labels. Every material statement about Lawie carries an evidence class. [IMPLEMENTED] in source at the basis commit; [TESTED] exercised by repository tests at that commit; [MEASURED] measured by an offline experiment of this suite on the stated public or synthetic data, never a production or customer measurement; [ANALYTICAL] follows by argument from stated assumptions; [SIMULATED] output of a stated model; [EXTERNAL] checked against a cited primary source; [PROPOSED] design or protocol not built or not run; [UNVALIDATED] a hypothesis, stated as such; [ABSENT] not implemented; [BLOCKED] awaits an external owner or decision. Statements describe the source repository at the basis commit, not a deployed service: no workflow described here has a dated production acceptance, nothing is independently audited, and no statement has been reviewed by an independent human expert. This is a technical publication, not legal advice, not an offer of services and not a statement that any use of Lawie complies with any law.

Doctrine, mission and position

> **Intelligence is not authority. Mission.** Make high-stakes institutional work executable by supervised machines: completed, approved, evidenced, defensible.
>
> **Doctrine.** Execution, not advice. No anchored source, no decisive claim. We support professionals; we render no legal judgment. The mission is a direction, not a present capability: section 10 states what exists today.

Legal AI has no shortage of fluent answers. It is short of answers that survive a reader who distrusts the vendor, the model and the operator, and who will open the Official Journal to check. Courts, regulators, auditors and opposing counsel are such readers. This paper is written for them, and so is the machinery it describes.

Six positions

Each position below is a design commitment. Lawie's source code meets some of them in part and fails others in measured ways, as the later sections report; each has a cost, stated with it.

1. **Intelligence is not authority.** Competence is a property of outputs. Authority is a property of their pedigree and of the mandate under which someone acts on them. A better model produces better candidates; it produces neither a source nor a permission. No model score, confidence value or self-description may stand in for an admitted source, an approval receipt or a signature. *Cost:* every consequential output waits on machinery that is narrower and slower than the model it constrains.
2. **Similarity is not support.** A statement can share almost every word with a provision and say the opposite: insert "not", exchange "shall" for "may", move a date, swap the controller for the processor. Overlap measures resemblance, and the legal effect of a sentence sits in the few tokens that overlap discounts. *Cost:* a verifier that checks the decisive elements rejects paraphrases a similarity score would pass, and still misses the error types it does not check (section 4).
3. **In force is not applicable.** Publication, entry into force, application and repeal are four facts, often different for different provisions of one act, and an amendment can move any of them. A system that knows only "in force" answers "does this apply today?" wrongly on every day between entry into force and application. *Cost:* per-provision schedules transcribed from the final provisions and re-verified whenever the law moves, as the AI Act's did in July 2026 (section 5).
4. **An approval is an act on exact content, not a flag.** An approval authorizes one tool, one version and one input, identified by its hash, within a time window, by a named person who is not the requester. A Boolean "approved" detached from content authorizes whatever is substituted for the content later. *Cost:* every change to an approved input, however small, needs a new approval, and long workflows accumulate review latency.
5. **A hash is not a witness, and integrity is not truth.** A hash chain shows that its records agree with each other. It does not show that none was cut from the end, or that the whole was not rewritten by someone able to recompute it; only an independent witness of the head closes those gaps. A record that is perfectly intact proves only what it records. *Cost:* evidence must carry its limitations, and anchoring needs a party the operator does not control (section 7).
6. **Refusal is an answer.** When the source is missing, the date unresolved, the approval absent or the verifier blind to the error type, the correct output is a refusal with its reason. A system scored for always answering learns to guess [1]. *Cost:* lower coverage, questions returned to a professional unanswered, and refusals that compound over the steps of a workflow (proposition 2.2).

Contributions

K1 A formal model of verifiable legal execution (sources, provisions, propositions, support, verifiers, release gates, authority chains, evidence chains) with three elementary results: the released error of a gate, the limit of what a verifier's error rates say beyond its error model, and the compounding of error and refusal over a workflow (section 2).

K2 LLRB-1, a pre-registered benchmark of verification components on eight EU acts, with every label fixed by construction from the official text rather than by a model (section 4; method in full in LL-BEN-01).

K3 Measured findings about Lawie's own components, negative results included: grounding that accepts every decisive edit, a confidence value that carries no information, a quotation matcher that is either blind or useless, an exact-span check that rejects typography, a temporal index that equates entry into force with applicability, and a citation rule that passes fabricated citations (sections 4 and 5; LL-REL-01, LL-ADV-01).

K4 BEN-02, an authorization benchmark of 552 scenarios labelled by explicit norms, and an evidence-integrity study of audit chains and signed dossiers (sections 6 and 7; LL-BEN-02, LL-EVP-01, LL-SEC-01).

K5 The AI Act amendment case: what Regulation (EU) 2026/1744 did to Article 113, how much schedule error each temporal heuristic carries, and how a legislative-diff engine fared on the amendment (section 5; LL-TMP-01, LL-RI-01).

K6 The Civic Guard evaluation programme: an accountable-publication design, a base-rate bound on screening alerts and five protocol-only studies that wait on external owners (section 8; LL-CG-01).

K7 An explicit evidence boundary: every material statement about Lawie carries an evidence class bound to a claim register, and the boundary is stated area by area (section 10 and Appendix B).

Where it stands

Nothing in this paper is shipped. No Lawie product has a dated production acceptance [ABSENT]; the production release is not deployed and awaits owner-controlled inputs [BLOCKED]; Civic Guard is quarantined [BLOCKED]; no independent audit, certification or expert review exists [ABSENT]; and the study that would measure errors in model-generated answers is defined and not run [PROPOSED]. What the paper offers is a model, a description of the source code at one commit, and measurements of components executed offline. The distance between that and a legal execution system a court could lean on is evidence that does not yet exist: error rates of generated answers, dated production acceptance, independent review. Every statement about Lawie here is bound to an entry in an internal claim register, so that the evidence, when it exists, can be compared with the paper statement by statement.

Contents

Abstract **1**

Doctrine, mission and position **2**

1 The problem **6**

1.1 Answers and acts 6

1.2 What the record shows 6

1.3 Why capability will not close the gap 6

1.4 Intelligence is not authority 7

1.5 Code is law, literally 7

1.6 What each mechanism establishes 7

2 A model of verifiable legal execution **8**

2.1 Sources, provisions and propositions 8

2.2 Support and decisive elements 9

2.3 Verifiers and release gates 9

2.4 Compounding 10

2.5 Authority chains 11

2.6 Evidence chains 12

3 Architecture **12**

3.1 Admitted sources and the versioned corpus 12

3.2 Retrieval 13

3.3 Verification 14

3.4 Agents, tools, grants and approvals 14

3.5 Tenant scoping 15

3.6 Audit and evidence 15

3.7 The model and the code 15

4 Verification: what is implemented and what LLRB-1 measured **15**

4.1 Design 16

4.2 Confirmatory results 17

4.3 Where lexical measures fail 18

4.4 Confidence 19

4.5 Quotations 19

4.6 Citations 20

4.7 Release discipline 20

4.8 What the measurements show 20

5 Legal time **21**

5.1 Four dates and two times 21

5.2 The AI Act's Article 113, rewritten 22

5.3 What the simplifications cost 23

5.4 What Lawie's temporal components do 23

5.5 Detecting the change itself 24

6 Authority: approvals, tenants and what an evaluator must see **24**

6.1 What the code implements 24

6.2 What BEN-02 measured 24

6.3 What an evaluator must receive 25

6.4 What it means 26

7 Evidence: chains, anchors and dossiers **26**

7.1 What a chain can show . . . 26
7.2 Signed dossiers . . . 27
7.3 Integrity is not truth . . . 27
8 Civic intelligence: accountable publication and the base rate . . . 27
8.1 Design . . . 28
8.2 The base rate . . . 28
8.3 What was measured and what was not . . . 28
9 Limitations . . . 28
9.1 What LLRB-1 does not measure . . . 29
9.2 Authorization and evidence studies . . . 29
9.3 Civic Guard . . . 29
9.4 The analytical results . . . 29
9.5 Independence and review . . . 30
9.6 Law and scope . . . 30
10 Where it stands . . . 31
11 Conclusion . . . 32
A Evidence classes . . . 32
B Index of the claims cited . . . 33
References . . . 35

1 The problem

1.1 Answers and acts

Most legal AI is built to answer. A user asks, a model writes, and the user decides what to do with the text. The allocation of risk is the one familiar from search engines: the system answers for nothing it says, and the user answers for everything done with it.

Legal work does not end in answers. Its products are acts: a filing that starts a limitation period, a notice that ends a contract, an approval that releases funds, a finding that names an institution. Each act has an author, a date, a legal basis and consequences for people who did not write it, and each must survive a reader who distrusts it. Opposing counsel, a regulator, an auditor or a court asks three questions of such an act: on what source does it rest, who had the right to do it, and can both be checked without trusting the party that produced it?

A system that produces acts must answer those three questions for every act it produces, and must decline to produce an act when it cannot. That is a different engineering problem from producing good answers. Progress on the second does not solve the first, and this paper is about the first.

1.2 What the record shows

The failure of answer-producing systems in law is documented. Dahl et al. [2] asked language models specific, verifiable questions about randomly drawn United States federal court cases and found hallucination rates between 58% (ChatGPT 4) and 88% (Llama 2). Magesh et al. [3] tested commercial legal research tools whose providers had described them as avoiding hallucination and found that each hallucinated on between 17% and 33% of queries; the providers' claims, they conclude, were overstated. The sanctions opinion in *Mata v. Avianca* records a brief that cited decisions which did not exist, produced by a chat assistant, and continued reliance on them after the court had questioned their existence [4]. The American Bar Association's Formal Opinion 512 states that a lawyer who uses generative AI remains bound by the duties of competence, confidentiality, communication, candour toward the tribunal, supervision and reasonable fees, may not rely uncritically on the tool's output, and must review it to a degree that depends on the tool and the task [5].

The opinion places responsibility on the professional. It does not say how a professional is to check a thousand citations, or the application dates of a regulation amended last month. The burden of verification stays with the human, and an answer-producing system does not reduce it; it moves the burden from writing to checking.

1.3 Why capability will not close the gap

It is tempting to treat these results as a stage of model development that better models will outgrow. Two results argue otherwise. Kalai and Vempala [6] prove that a calibrated language model must hallucinate on "arbitrary" facts, those whose truth cannot be inferred from regularities in the training data, at a rate close to the fraction of such facts that occur exactly once in training. The bound does not reach facts that recur often in training data, but the case name, docket number, pinpoint citation or wording of a rarely cited decision is an arbitrary fact in exactly this sense. Kalai et al. [1] argue further that hallucination persists because training and evaluation reward guessing over the acknowledgement of uncertainty: a model optimised to score well on benchmarks that give no credit for abstaining learns to guess. Their remedy is a change in what is rewarded, not a better model.

The consequence for law is direct. The error that matters, a confident, specific and wrong statement about what a source says or when it applies, is not a defect that scale removes; it follows from the objective.

A system that must not make it needs machinery outside the model: access to the source, a check of the statement against the source, and the freedom to refuse.

1.4 Intelligence is not authority

Law has a precise vocabulary for the difference between being right and having the standing to be relied on. In Hart's account a rule belongs to a legal system because it satisfies the criteria of the system's rule of recognition, criteria that officials in fact apply; the criteria he describes are typically matters of pedigree, such as enactment by a competent body, although his account allows them to include tests of content [7]. Raz's sources thesis states the strict form of the point: the tests for identifying the existence and content of law depend on social facts, such as enactment, publication and decision, and are applied without resort to moral argument [8].

Two consequences follow for machines. First, the warrant of a legal proposition is its source. A statement about what a regulation requires is worth what the regulation, in its applicable version, says. A model's fluency, consistency or benchmark score adds nothing to that warrant. Second, the right to act on a legal position is conferred, not inferred. A clerk of great ability has no authority to file until a principal gives it; ability is a reason to delegate, never a substitute for delegation.

We call the pair, the warrant to be believed and the right to act, *authority*, and the capacity to produce good candidates for either *intelligence*. The doctrine of this paper is that intelligence is not authority. It is a claim about kinds, not about current model quality: no improvement in a model converts its output into a source or its proposal into a mandate. Corollary 2.3 states the point formally.

1.5 Code is law, literally

Lessig [9] observed that in networked systems architecture regulates conduct as effectively as law, and often more so, because it fixes what is possible before any rule is applied. For a system that executes legal work the observation is literal. If the binding of an approval to its content, the separation of requester and reviewer, the date on which a provision applies or the refusal to cite an unadmitted source is not enforced in code, it is not a property of the system, whatever the system's policies say. Making law executable means turning those requirements into constraints a machine cannot route around, and measuring where it can.

1.6 What each mechanism establishes

Table 1 separates seven properties of a legal output that a legal-execution system must keep apart, and states for each what can establish it and what it leaves open.

The characteristic error is to move down the table: to present relevance as support ("grounded in sources"), the presence of a citation as authority, an audit log as evidence of correctness. Each move is a category error, and sections 4 to 7 measure what each one costs in Lawie's own components.

> **Thesis.** Legal AI cannot be made trustworthy by making it more intelligent. It can be made accountable by making authority executable: every decisive proposition anchored to an admitted official source applicable on the relevant date, every consequential action bound to an approval by a named person on exact content, every artifact carrying evidence a hostile party can check together with the limits of that evidence, and a refusal wherever any of these is missing. This paper models that machinery, describes how far Lawie's source code implements it, and measures where the implementation falls short.

Table 1: What each mechanism establishes about a legal output, and what it leaves open.

Property	Question it answers	What can establish it	What it does not establish
Fluency	Does the text read like competent legal writing?	A language model	Anything about correctness
Relevance	Does a retrieved source concern the same matter?	Lexical or embedding similarity, ranking	That the source says what the statement says
Support	Does the cited provision fix every decisive element of the statement?	A check of those elements against the provision text	That the provision applies on the relevant date
Applicability	Does the provision apply on this date, to these facts?	A per-provision schedule transcribed from the final provisions; the facts	That the provision was read correctly
Authority to act	Who was entitled to take this step, on what content?	An approval receipt bound to tool, version, input, time window and a named person	That the step was wise or lawful
Integrity	Are these the bytes that were recorded?	A hash compared with an independently witnessed value	That the bytes are true, complete or admissible
Correctness	Is the legal position stated right?	Accountable professional judgment	Certainty: it remains contestable

2 A model of verifiable legal execution

This section names the objects a legal-execution system handles and the relations it must decide, precisely enough that a component can be tested against them. The results are elementary: Bayes' rule, a union bound and an argument about the inputs of a decision. They are stated because the way legal AI is described routinely contradicts them. Every result here is analytical: it holds under its stated assumptions, whatever an implementation does, and it is labelled accordingly.

2.1 Sources, provisions and propositions

Definition 2.1 (Admitted source corpus)**.** An *official source* is a publication of an issuing authority with a stable locator, such as the Official Journal of the European Union. An *observation* of a source records one retrieval: the locator, the retrieval time, the payload bytes, their hash $\mathcal{H}(\text{payload})$ and the licence under which the payload may be used. The *admitted corpus* $\mathcal{S}$ is the set of observations retrieved under an admission decision, taken by an accountable person, that records the source's official status, its jurisdiction and the rights that apply to it.

Definition 2.2 (Provision)**.** A *provision* is a tuple $s = (o, a, [b, e), I)$: an observation $o \in \mathcal{S}$, an address $a \in \mathcal{A}$ (instrument, article, paragraph, point), a byte interval $[b, e)$ of the payload of o, and a validity record I that holds the dates of publication, entry into force, application and repeal (section 5). $\mathsf{text}(s)$ is the byte string at $[b, e)$. Two provisions with the same address and different observations or intervals are different *versions*.

Definition 2.3 (Proposition)**.** An output asserts a finite set Φ of propositions. A proposition $\varphi \in \Phi$ carries a citation $c(\varphi) \in \mathcal{A}$ and an as-of date t_q, the date the proposition is about. A proposition is *decisive* for its output if changing its truth value changes what the output tells its reader to do or to believe about a legal position.

2.2 Support and decisive elements

Definition 2.4 (Decisive elements)**.** The *decisive elements* of a legal proposition are the components whose alteration alters its legal effect: the actor; the deontic operator and its polarity (*shall*, *may*, *shall not*); quantities, dates and time units; cross-references; the conditions and exceptions that bound it; and the provision to which it is attributed. A *decisive edit* changes exactly one decisive element.

Definition 2.5 (Support)**.** A provision s *supports* a proposition φ at t_q, written $\mathsf{Supp}(s, \varphi, t_q)$, if and only if (i) the address of s is $c(\varphi)$; (ii) s is applicable at t_q; and (iii) every decisive element of φ is fixed by $\mathsf{text}(s)$, read together with the definitions and cross-references that s itself invokes.

Remark 2.1. Support is a legal-semantic relation. We know of no procedure that decides condition (iii) for unrestricted natural language, and we claim none. Every verifier in this paper approximates Supp, and the approximation error is what section 4 measures. Conditions (i) and (ii) are different in kind: given admitted sources and transcribed schedules they are mechanical, which is why a system can and should get them exactly right.

2.3 Verifiers and release gates

Definition 2.6 (Verifier)**.** A *verifier* is a computable function $V : (\varphi, s) \mapsto \{\mathsf{ACCEPT}, \mathsf{REJECT}\}$. On a distribution D of proposition–provision pairs its *false-accept rate* is $\mathrm{FAR}_D(V) = \Pr_D[V = \mathsf{ACCEPT} \mid \neg\mathsf{Supp}]$ and its *false-reject rate* is $\mathrm{FRR}_D(V) = \Pr_D[V = \mathsf{REJECT} \mid \mathsf{Supp}]$. Both depend on D; no single number characterises a verifier.

Definition 2.7 (Release gate)**.** A *release gate* G releases an output if its verifier accepts every decisive proposition against the provision the proposition cites; otherwise it marks the output WITHHELD and records the reason. Its *coverage* cov is the probability that an output is released, and its *selective risk* risk is the probability that a released output carries a decisive error [10, 11].

Proposition 2.1 (Released error of a gate)**.** *Let a population of outputs carry a decisive error with probability $e \in (0, 1)$, and let the gate accept an output with a decisive error with probability F and withhold a correct output with probability $R < 1$. (F and R are rates of the gate on outputs; they equal the verifier's* FAR *and* FRR *only for outputs with one decisive proposition.) Then*

$$\mathrm{cov} = eF + (1-e)(1-R), \qquad \mathrm{risk} = \frac{eF}{eF + (1-e)(1-R)}, \qquad \frac{\mathrm{risk}}{1-\mathrm{risk}} = \frac{e}{1-e} \cdot \frac{F}{1-R}. \tag{1}$$

Proof. Let E be the event that an output carries a decisive error and A the event that the gate accepts it. Then $\Pr[A] = \Pr[A \mid E]\Pr[E] + \Pr[A \mid \neg E]\Pr[\neg E] = Fe + (1-R)(1-e)$, and $\mathrm{risk} = \Pr[E \mid A] = \Pr[A \mid E]\Pr[E]/\Pr[A]$ by Bayes' rule. The odds form is the ratio of $\Pr[E \cap A] = eF$ to $\Pr[\neg E \cap A] = (1-e)(1-R)$. □

The odds form is the useful one: a gate multiplies the prior odds of a decisive error by $F/(1-R)$ and does nothing else, so the released error depends on all three of e, F and R. The coverage formula also holds for $e = 0$. The identity is the register's analytical claim [ANALYTICAL]. Two corollaries follow.

Corollary 2.1 (A verifier that accepts every error cannot reduce it)**.** *If $F = 1$, then* $\mathrm{risk} = e/(e + (1-e)(1-R)) \geq e$*, with equality if and only if $R = 0$. A gate whose verifier accepts every decisive error releases outputs at least as erroneous as the population it is given, whatever it is called and however its outputs are labelled.*

Proof. Substitute $F = 1$ in (1); the denominator is at most 1, and equals 1 exactly when $R = 0$. □

Corollary 2.2 (A verifier's assurance extends exactly as far as its error model)**.** *Partition decisive errors into types $j \in J$ with population shares w_j ($\sum_j w_j = 1$) and type-wise false-accept rates F_j, so that $F = \sum_j w_j F_j$. Let $M \subseteq J$ be the types on which F_j is known, $F_M = \sum_{j \in M} w_j F_j$ and $W_U = \sum_{j \notin M} w_j$. If nothing is known about F_j for $j \notin M$, the selective risk can take every value in $[\rho(F_M), \rho(F_M + W_U)]$, where $\rho(F) = eF/(eF + (1-e)(1-R))$.*

Proof. As the unknown F_j range over $[0,1]$, F ranges over $[F_M, F_M + W_U]$. For $0 < e < 1$ and $R < 1$, $\rho'(F) = e(1-e)(1-R)/(eF + (1-e)(1-R))^2 > 0$, so ρ is continuous and increasing and attains every value between the endpoints. □

Corollary 2.2 is the formal reason why a benchmark that omits an error type says nothing about a verifier's behaviour on it, and why "validated" must always name the error types validated against.

Worked example 2.1 (Illustrative arithmetic)**.** Suppose, for illustration only, that each output carries one decisive proposition, that one output in ten carries a decisive error ($e = 0.10$), that the verifier never rejects a supported output ($R = 0$), and that the population's decisive errors are distributed like the 1,959 decisive edits of LLRB-1 (section 4). A verifier that accepts every decisive error releases outputs with a 10% error rate, exactly the rate it was given (corollary 2.1). The best calibrated lexical verifier in LLRB-1, which accepted 27.1% of those edits, releases outputs with a 2.9% error rate at 92.7% coverage. The typed prototype, which accepted none of the modelled edits and 456 of the 515 held-out ones, accepted 456 of 1,959 decisive edits (23.3%) under that mix and releases 2.5% erroneous outputs at 92.3% coverage. Under another mix its result moves anywhere between 0% (no unmodelled errors) and 9.0% (only unmodelled errors), as corollary 2.2 says; the LLRB-1 mix is a design choice of the benchmark, not a property of any population of answers. The base rate and the type mix of decisive errors in generated legal answers are unmeasured: the study that would measure them is defined and not run [PROPOSED]. This is why the paper reports verifiers' error rates and not released error rates.

Remark 2.2 (Refusal inside an output)**.** An output with m decisive propositions, checked independently by a verifier with per-proposition false-reject rate R_1, is released only if all m are accepted. A correct output is then withheld with probability $R = 1 - (1 - R_1)^m$: false rejection compounds within an output exactly as it compounds across the steps of a workflow (proposition 2.2).

2.4 Compounding

Proposition 2.2 (Compounding of step reliability)**.** *A run has k steps; let C_i be the event that step i is decisively correct, with $\Pr[C_i] = q_i$.*

(a) Without any assumption about dependence, $1 - \sum_{i=1}^{k}(1 - q_i) \le \Pr[\bigcap_i C_i] \le \min_i q_i$.

(b) If the events are independent and $q_i = q$, then $\Pr[\bigcap_i C_i] = q^k$.

(c) If, in addition, each step passes an independent gate with error rates (F, R) at error probability $e = 1 - q$, a run clears every gate with probability cov^k, and a run that clears every gate is correct at every step with probability $(1 - \mathrm{risk})^k$, with cov and risk as in (1).

Proof. (a) By the union bound, $\Pr[\bigcup_i \neg C_i] \le \sum_i (1 - q_i)$; and $\bigcap_i C_i \subseteq C_j$ for every j. (b) Independence. (c) Apply proposition 2.1 to each step and multiply, using independence across steps. □

Two lessons follow. The first is familiar: a workflow of twenty steps that are each 99% reliable is wrong somewhere in about one run in five, so agentic legal work needs per-step reliability far beyond what a single answer needs. The second is less often stated. Gating converts errors into halts, and halts compound too. With $R = 0.05$, a twenty-step gated run whose every step is correct passes each gate with probability 0.95 and halts somewhere with probability $1 - 0.95^{20} \approx 0.64$. When errors are rare, the human-review load of

Table 2: End-to-end correctness q^k of a run of k independent steps, each correct with probability q (proposition 2.2(b)). Arithmetic, not measurement.

Step reliability q	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 50$
0.999	0.999	0.995	0.990	0.980	0.951
0.99	0.990	0.951	0.904	0.818	0.605
0.95	0.950	0.774	0.599	0.358	0.077
0.90	0.900	0.590	0.349	0.122	0.005

a long gated workflow is set by the false-reject rate, not by the error rate. A verifier for agentic work must therefore keep both F and R low; a verifier that buys a low F with a high R returns most runs to a human. Refusal is an answer (Position 6), and it is not free.

The bounds in (a) need no independence; the product in (b) does. Without independence only the union bound survives: it bounds end-to-end error below δ only if the per-step errors sum to less than δ, which for equal steps means per-step error below δ/k. Positively correlated errors can leave end-to-end correctness as high as $\min_i q_i$, and negatively correlated ones push it down to the union bound.

2.5 Authority chains

Definition 2.8 (Authority chain)**.** A *principal* P is an organisation on whose behalf work is done. A *named human authority* h holds a role under P. An *action* is $\alpha = (T, v, x)$: a tool T, its version v and its input x, requested by a requester u (a person, or an agent acting for one). An *approval receipt* is $\mathsf{appr} = (T, v, \mathscr{H}(x), u, h, t_{\text{req}}, t_{\text{rev}}, t_{\text{exp}}, d)$ with decision $d \in \{\text{approved}, \text{rejected}, \text{pending}\}$. For a set $\mathscr{R}$ of receipts, the authorization predicate $\mathsf{Auth}(\alpha, \mathscr{R}, t)$ holds at time t if and only if $\mathscr{R}$ contains a receipt appr with:

(i) *binding*: appr names exactly $(T, v, \mathscr{H}(x))$;
(ii) *validity*: d is approved, h is named, and $t_{\text{req}} < t_{\text{rev}} \le t < t_{\text{exp}}$;
(iii) *separation*: $h \neq u$;
(iv) *role*: h holds under P a role that covers T;
(v) *quorum*: where the policy for T requires n approvals, $\mathscr{R}$ contains n receipts that each satisfy (i)–(iv), with pairwise distinct reviewers.

If an agent a starts a sub-agent a', written $a \rightsquigarrow a'$, then $C(a') \subseteq C(a)$ and every approval requirement that binds a binds a'. The *authority chain* of an action is $P \to h \to \mathsf{appr} \to \alpha \to$ artifact; the artifact records the receipt, and the receipt records the chain.

The definition formalises Lawie's own authority model, which binds every consequential action to a principal, a named human authority, a scoped and expiring approval receipt, the action and its artifact, and keeps approval, execution and finality as distinct states; it is a design, and section 3 describes the controls that implement it [PROPOSED]. The predicate has five conjuncts, and each guards against a distinct failure: substitution of content (i), stale or manufactured consent (ii), self-dealing (iii), approval by someone without standing (iv) and unilateral action where policy demands more than one person (v). Separation of duty is central to the commercial integrity policies analysed by Clark and Wilson [12], and the requirement that every access be checked against current authority is the complete mediation of Saltzer and Schroeder [13].

Lemma 2.1 (A check sees only its inputs)**.** *Let a norm $N : X \times Y \to \{allow, deny\}$ be enforced by a decision procedure $D : X \to \{allow, deny\}$ that receives only $x \in X$. If $N(x, y) \neq N(x, y')$ for some x, y and y', then D disagrees with N on (x, y) or on (x, y').*

Proof. $D(x)$ takes a single value, and N takes two different values on the two pairs. □

The lemma is trivial, and it decides several questions of design. An approval evaluator must receive the requester's identity to enforce separation (iii); a tool-authorization function must learn whether a request was induced by untrusted content to refuse it on that ground; a child's authorization must receive its parent's restrictions to enforce delegation. These are requirements on the interfaces of any authorization layer, and section 6 turns them into the design of a benchmark. Applied to models, the lemma gives the formal core of the doctrine.

Corollary 2.3 (Intelligence is not authority)**.** *Let x be the content of a run, meaning the proposed action together with every model output, including any score, confidence value or self-description, and let y be the receipts, identities and times on which* Auth *depends. For every proposed action there are values of y under which it is authorized and values under which it is not. Hence every procedure that decides authorization from x alone errs on some history, however good the outputs in x are.*

Proof. Take y containing a set of receipts that satisfies (i)–(v) and y' containing no receipt; Auth differs, and lemma 2.1 applies. □

As mathematics the corollary is empty; as design it is a prohibition. The quality of an output, its accuracy, its confidence or the model's account of its own authority, is never an input to an authorization decision. Lawie's citation-authority predicate is built this way: model output is never one of its inputs [TESTED].

2.6 Evidence chains

Definition 2.9 (Evidence chain and witness)**.** A chain $\mathscr{C} = (r_1, \ldots, r_n)$ has heads $h_0 = \mathsf{GENESIS}$ and $h_i = \mathscr{H}(h_{i-1} \,\|\, r_i)$; it *verifies* if every h_i recomputes from the records. A *witness* of the head is a party independent of the writer that holds (n, h_n) and a time at which it received them, such as a time-stamping service or a public transparency log [14, 15].

A chain is evidence of internal consistency; a witnessed head is evidence of what existed at a time. Section 7 states what each can show and what the evidence-integrity study measured.

3 Architecture

Lawie's source repository implements most elements of the model in some form. This section describes the components by function, not by name, and states for each what the source does at the basis commit; the label after each statement gives its evidence class. Component names, parameters other than those that are themselves findings, and deployment details are not part of this edition. The statements describe the source, not a deployed service, and none is a security assurance. Figure 1 shows the two paths of authority, the warrant to be believed and the right to act, and the evidence both must leave behind.

3.1 Admitted sources and the versioned corpus

A source enters the corpus as an observation that records its locator, the retrieval time, payload and content hashes, an object snapshot, the versions of the connector and parser that produced it, the licence revision under which it may be used, and an immutable chain to any later observation that supersedes it [IMPLEMENTED]. A source counts as official only under an approved, active source contract that fixes its official type, its jurisdiction and its reviewed licence [TESTED]. Admission is a human decision, and none has been taken for production: the production connector profile lists twelve official sources, none admitted, because no source contract is approved [BLOCKED].

A provision version is authoritative only if its text is an exact UTF-8 byte span of the payload of an observation retrieved under such a contract, derived by a reviewed span derivation and proven against the

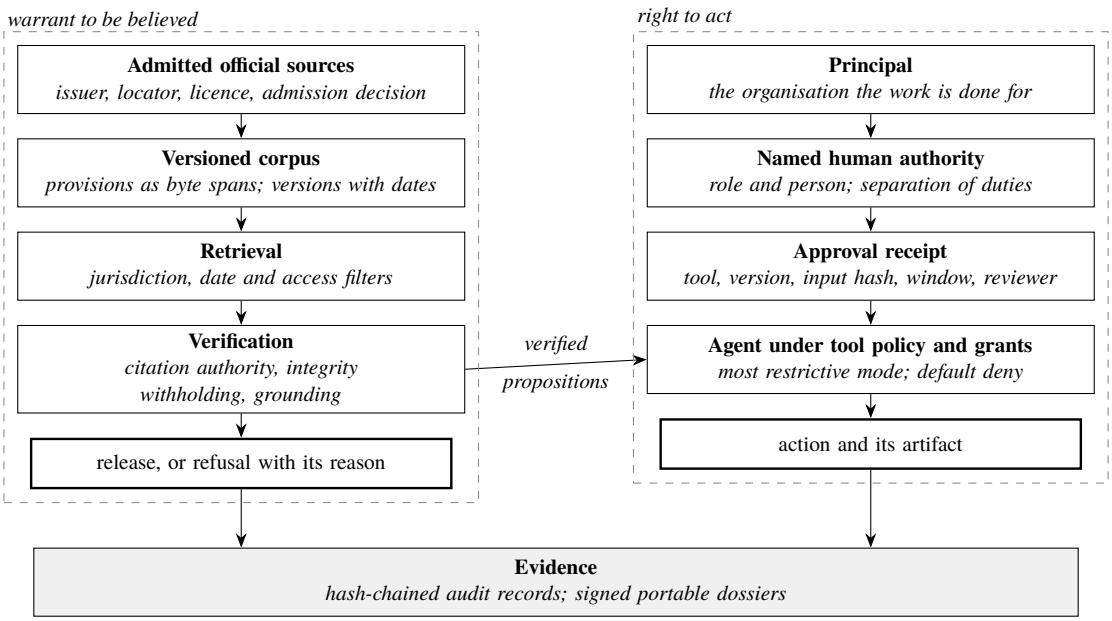

Figure 1: The two paths of authority in Lawie's architecture and the evidence they leave. Boxes name functions, not components.

stored payload by hash, if that observation is still the current state of the official record, and if the version is in force at the citation's date [TESTED]. Exactly one span derivation is admitted, for paragraphs of the Czech official collection of laws; no EU, Slovak or case-law text can at present be verified as authoritative [IMPLEMENTED].

Versions carry half-open validity intervals, and a temporal index returns the version in force at a date [TESTED]. The EUR-Lex parser starts a version at the entry-into-force date and leaves its end open [IMPLEMENTED]; staggered dates of application, transitional provisions and consolidated versions are not modelled [ABSENT]. Section 5 measures what that costs.

3.2 Retrieval

Retrieval combines dense and sparse search, applies jurisdiction, temporal and access filters, reranks, and drops citations that no retrieved source backs [IMPLEMENTED]. Jurisdiction routing follows a declared chain (Czech, EU, European Convention on Human Rights, international) and drops documents without a jurisdiction tag from jurisdiction-routed results [TESTED]. A missing-document detector reports expected sources absent from the retrieved set; it does not refuse the answer [TESTED]. Fusion weights and ranking constants are not disclosed.

3.3 Verification

Four mechanisms stand between a generated statement and its release.

1. *Citation authority.* The predicate described above decides whether a cited provision text is authoritative. Every refusal carries a typed reason, among them a missing or unapproved source contract, an unreviewed licence, a superseded record, a span or hash mismatch and a version not in force at the cited date [TESTED]; model output is never one of the predicate's inputs [TESTED].
2. *Integrity withholding.* A research answer with any citation that no retrieved source backs is withheld, blocks reliance and requires human review [TESTED].
3. *Citation gate.* Agent outputs require factuality and citation-accuracy metrics; a missing or failing metric marks the output contradicted, fails the run and withholds the output [TESTED].
4. *Grounding.* A staged verification pipeline includes a lexical grounding check that compares a claim with the retrieved text it cites; section 4 measures what it accepts.

Three properties shape what these mechanisms deliver. Outside production the pipeline defaults to a degraded-review policy, under which an unavailable model service yields a stage that is "not run" and does not fail the output; its production configuration defaults to fail-closed [IMPLEMENTED]. No component produces a calibrated probability of correctness [ABSENT]. And case-law good-law checks cannot pass: the validity service returns no determination, and the gate requires a positive one [IMPLEMENTED]. The gateway adds two response rules for legal conclusions, a jurisdiction-scope rule that treats Czech, Slovak and EU law as one compatible bloc [IMPLEMENTED] and a citation-required rule; section 4 reports how both behaved.

3.4 Agents, tools, grants and approvals

Lawie registers 81 agent types [TESTED]. An agent acts only through tools. A tool call's authority is the most restrictive of its capability modes (automatic, allowed within limits, requires approval, forbidden), and an approval-gated mode needs an approval for the exact call [TESTED]; the most restrictive composition of no modes at all is "forbidden", and the decision schema rejects an allowed decision in that mode or an approval-gated one without approval evidence [TESTED]. The agent runtime pauses for an exact-input approval before any plan step that uses a privileged action tool or a tool of the highest risk tier, whatever the tool's own approval flag [TESTED]. The tenant tool-grant decision is default-deny: without an active grant the call is refused, and the decision also checks data-class and risk ceilings, the tenant, run and user identities, the agent's admission ceiling and the persisted approval for the input [TESTED]; it reads current state on every call and re-checks the grant's expiry after its reads and after its mandatory audit write [IMPLEMENTED].

An approval authorizes only the tool identity, its exact semantic version and the SHA-256 of the canonical current input (a requirement may name another tool and version, but the input digest must still match, and the core grant decision requires the same tool); a referenced hash can narrow but never replace the current input's hash [TESTED]. An approval is usable only if it is approved, was reviewed by a named reviewer after it was requested and before it is used, and has not expired; composite requirements (single, any of, all of, sequence) are evaluated, and a malformed policy is invalid, never approved [TESTED]. An approval decision must come from an authenticated human reviewer who holds the approve permission and is within the run's matter scope; internal service calls and API-key callers are refused [IMPLEMENTED]. The core service accepts an approval request only with a bounded lifetime and refuses to reuse a gate identifier for a different run, tool, version, input or lifetime [IMPLEMENTED], and provider writes consume each approval exactly once on dispatch, re-checking the current grant and the reviewer's authority [IMPLEMENTED].

3.5 Tenant scoping

Requests to the agent orchestrator must carry a signed identity assertion bound to the method, path and body; tenant or user substitution is rejected, and state-changing assertions are single-use [TESTED]. Without a verification key, or for a state-changing request without a replay store, the orchestrator executes nothing [TESTED]. Tenant scoping compares the requested tenant with the authenticated tenant, for administrators too [TESTED].

3.6 Audit and evidence

Agent runs leave hash-chained audit records whose verifier section 7 tests. Signed portable dossiers of institutional execution are signed with an asymmetric key held in a managed key-management service [IMPLEMENTED], and verify offline against pinned key fingerprints (Ed25519, ECDSA P-256, RSA-PSS) [TESTED]. A dossier manifest is valid only if it states, inside the signed manifest, that verification proves neither substantive truth nor admissibility nor legal correctness, and creates no authority [IMPLEMENTED].

3.7 The model and the code

Table 3 sets each element of the model beside the function that implements it. The predicates that check pedigree and binding exactly (span hashes, approval hashes, tenant identifiers) are built and tested; the relations that need judgement about meaning or time (support and applicability) are approximated, and sections 4 and 5 measure the cost.

Table 3: Elements of the model (section 2) and the functions that implement them at the basis commit.

Model element	Implementing function	State
Admitted source (definition 2.1)	Source contract with reviewed licence; observation record	Built [IMPLEMENTED]; no production source admitted [BLOCKED]
Provision (definition 2.2)	Byte-span derivation proven by hash	Tested [TESTED]; one derivation admitted [IMPLEMENTED]
Validity record	Temporal index of versions	Tested [TESTED]; entry into force only [IMPLEMENTED]
Support (definition 2.5)	Lexical grounding check	Accepted every decisive edit in LLRB-1 [MEASURED]
Release gate (definition 2.7)	Integrity withholding; citation gate	Tested [TESTED]
Refusal with reason	Typed refusal reasons of the authority predicate	Tested [TESTED]
Authority chain (definition 2.8)	Approval evaluator; grant decision; authenticated reviewer	Tested [TESTED]; built [IMPLEMENTED]; aggregate outcome in section 6
Evidence chain (definition 2.9)	Hash-chained audit records; signed portable dossiers	Measured [MEASURED]; tested [TESTED]

4 Verification: what is implemented and what LLRB-1 measured

Section 3 listed the verification mechanisms. The question that matters is whether they detect the errors that change legal meaning. LLRB-1, the Lawie Legal Reliability Benchmark, asks it of the components themselves, on official EU text, with labels that no model assigned. LL-BEN-01 reports the method in full, LL-REL-01 places the results in the wider argument about reliability, and LL-ADV-01 reads them adversarially. This section gives the design and the findings.

LLRB-1 differs in its object from the legal benchmarks that measure models, such as LegalBench, CaseHOLD and LexGLUE [16–18], and from attribution measures that score generated text against sources [19–

22]: it measures the components that decide whether a statement is supported. Its method is the contrast set, a minimal edit that changes the correct label [23, 24], used as a behavioural test [25] of whether a verifier relies on the heuristic that McCoy et al. [26] found in inference models: lexical overlap taken for entailment.

4.1 Design

Corpus. The corpus holds 3,028 provisions of eight EU acts retrieved from EUR-Lex on 26 November 2026; the article count of each act equals the act's own [MEASURED]. Table 6 in section 5 lists the acts and their dates. The texts are a research copy; only the Official Journal is authentic.

Items. Statements are sentences drawn from the acts. For each statement the benchmark builds its verbatim form, at most one typographic variant (a no-break space, other quotation marks, an initial capital or final punctuation), one decisive edit for each edit family that applies, and a misattribution: the same text cited to the most lexically similar other provision of the same act. The support family has 5,820 items built from 1,007 statements, 4,053 of them in the test split, and every label is fixed by construction; none is assigned by a model [MEASURED]. Splitting by article keeps every article in one split only. No lawyer has reviewed the labels [ABSENT]; construction makes them mechanical, not infallible, and items whose edited text the provision itself contains are removed.

Edit families. Six families of decisive edit are *modelled* by the typed prototype described below: negation (insert or remove *not* after the first deontic modal), modality (*shall* and *may* exchanged, *must* weakened to *may*), quantity, date (a year shifted), time unit (*days* and *months* exchanged) and cross-reference (an article, paragraph or point number changed). Three are *held out*: actor (two legal roles of the act exchanged, such as controller and processor), comparative or quantifier (*at least* and *at most* exchanged, *all* weakened to *some*) and deleted exception (a clause beginning *unless*, *except*, *other than* or *provided that* removed). Every edit is constructed to change the legal effect of the statement; none is a paraphrase.

Systems. Lawie's components were imported unchanged from the basis commit and executed offline; each run records the commit and the SHA-256 of each component file it loads [MEASURED], and the runners refuse to run when one of those files differs from the commit; the files those components import are not checked [IMPLEMENTED]. Lawie's lexical grounding check receives the cited provision as its only retrieved text. Beside the components the benchmark runs reference baselines: byte-exact and canonical substring matching, and three lexical measures, namely token containment, token F1 and Jaccard similarity, each calibrated on the development split to keep 95% of supported statements. To check the harness, the component's decisions were reproduced independently on 4,053 of 4,053 test items [MEASURED]. The typed prototype accepts a statement only when it is almost entirely contained in a clause window of the provision and its modelled decisive elements (deontic operator with polarity, quantities, dates, time units and cross-references) equal the window's. It is a research instrument of this suite, not a Lawie component [PROPOSED].

Registration. The hypotheses, thresholds and test-split checksums were committed before the confirmatory run, which ran once [MEASURED]. The registration followed exploration: the development split was used to build the harness, set the thresholds and choose the hypotheses, and an earlier hypothesis, that every lexical baseline accepts more than half of the decisive edits, was refuted for the calibrated baselines on development data and replaced; the registered H2 tests a bar of 5%, not 50%, and the registration lists no deviation [MEASURED]. Statements were sampled per act and enriched with statements to which rarer edit families apply, so pooled rates depend on that mix and the primary results are per family [IMPLEMENTED]. Pre-registration protects the test from tuning, not the choice of what to test [27]. Tests are exact binomial tests, an exact McNemar test for the paired comparison [28] and Holm's correction where a hypothesis

comprises several tests [29]; intervals for the hypotheses are exact Clopper–Pearson intervals [30], and descriptive rates carry Wilson intervals [31].

4.2 Confirmatory results

Table 4: LLRB-1 confirmatory hypotheses on the test split. Counts are items accepted over items; intervals are 95%; p-values are exact and Holm-corrected where a hypothesis comprises several tests. All four hypotheses were supported.

Hypothesis and quantity	Count	95% CI (%)	Test	p (Holm)	Result
H1 decisive edits accepted by Lawie grounding	1,959/1,959	99.8–100.0	binomial > 0.5	$< 10^{-6}$	supported
H1 misattributions accepted by Lawie grounding	698/698	99.5–100.0	binomial > 0.5	$< 10^{-6}$	supported
H2 decisive edits accepted, calibrated containment	531/1,959	25.1–29.1	binomial > 0.05	$< 10^{-6}$	supported
H2 decisive edits accepted, calibrated token F1	1,062/1,959	52.0–56.4	binomial > 0.05	$< 10^{-6}$	supported
H2 decisive edits accepted, calibrated Jaccard	1,015/1,959	49.6–54.0	binomial > 0.05	$< 10^{-6}$	supported
H3a modelled edits accepted, typed prototype vs. containment	0 vs. 392 of 1,444	FRR 0% (0.00–0.26)	McNemar; binomial < 0.05	$< 10^{-6}$	supported
H3b held-out edits accepted, typed prototype	456/515	85.5–91.2	binomial > 0.5	$< 10^{-6}$	supported

H1: similarity is not support. Lawie's grounding check accepted 1,959 of 1,959 statements carrying one decisive edit and 698 of 698 statements cited to the wrong provision, with 95% lower bounds of 0.998 and 0.995 [MEASURED]. It does not distinguish "the provision says X" from "the provision says not X". By corollary 2.1, a gate built on it cannot lower the decisive-error rate of what it releases, on any population whose errors resemble these edits.

H2: calibration does not rescue similarity. Calibrated on the development split, with each threshold at the 5th percentile of supported items, containment, token F1 and Jaccard similarity accepted 27.1% (95% CI 25.1–29.1), 54.2% (52.0–56.4) and 51.8% (49.6–54.0) of decisively edited statements [MEASURED]. The failure is structural, not a matter of threshold. A decisive edit changes one or two tokens, which moves an overlap score about as far as a harmless typographic difference does, and a threshold that keeps almost every correct statement cannot separate the two. Containment, the strictest of the three (its calibrated threshold requires every token of the statement to occur in the provision), fails differently: it cannot see a deletion at all (proposition 4.1), and a substituted word that occurs elsewhere in the provision passes.

H3: typing protects exactly what it types. The typed prototype accepted none of 1,444 edits of the six modelled types and rejected none of 1,396 supported statements, with a 95% upper bound of 0.26% on false rejection [MEASURED]. The first half of that result is close to true by construction, since the prototype compares exactly the elements those edits change; the informative half is the absence of false rejections. On the three held-out types it accepted 456 of 515 edits (88.5%, 95% CI 85.5–91.2) [MEASURED]. Typed verification removes the errors whose type it checks and no others, as corollary 2.2 requires.

4.3 Where lexical measures fail

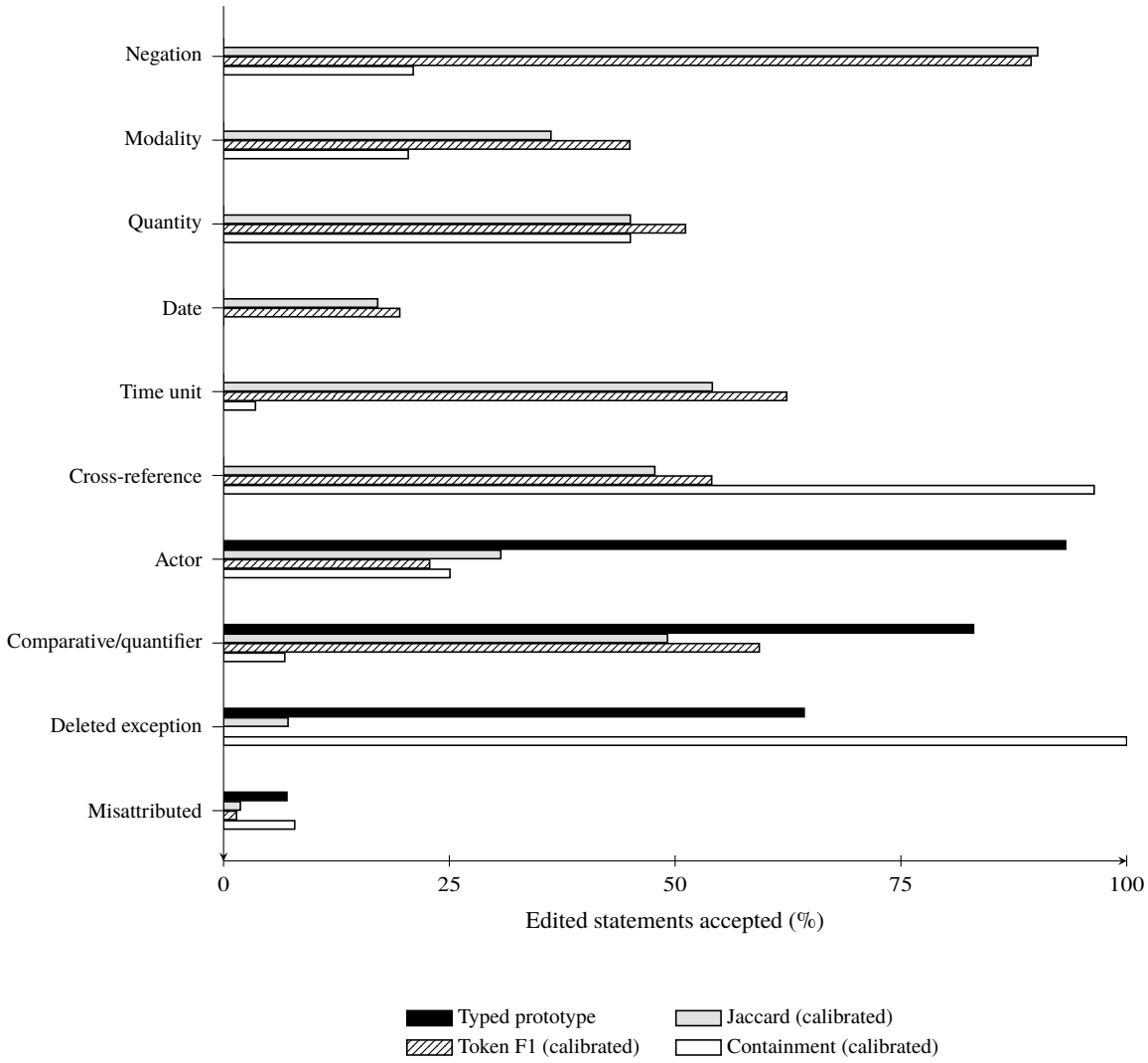

Figure 2: Share of decisively edited and misattributed statements accepted, by edit family, for the three calibrated lexical verifiers and the typed research prototype on the test split. Lawie's grounding check accepted 100% in every family and is not drawn. The prototype accepted no edit in the six families it models (bars of zero length); actor, comparative or quantifier and deleted exception are held out from it.

Figure 2 breaks the result down by family. Each lexical measure has a blind spot of its own. Token F1 and Jaccard accepted about nine in ten negations, because inserting or removing one short word barely changes the overlap. Containment rejected most edits that bring in a word the provision lacks, such as a shifted year or another time unit, and accepted every deleted exception, which the following analytical result explains.

Proposition 4.1 (Containment cannot see a deletion)**.** *Let containment accept a statement when the share of its tokens that occur in the provision reaches a threshold. If every token of φ occurs in* $\mathsf{text}(s)$*, then every token of any statement obtained by deleting words from φ also occurs in it, and containment accepts the shortened statement at every threshold.*

Proof. The tokens of the shortened statement are a sub-multiset of those of φ, each of which occurs in $\mathsf{text}(s)$; its containment is therefore 1. □

Deleting the exception from "shall … unless …" turns a qualified duty into an unqualified one. It is pure deletion, and containment cannot detect it [ANALYTICAL]. The same holds for a negation edit that removes a *not*. The typed prototype fails on the held-out families for the opposite reason: it checks only the element types it models, and actor, comparative and exception are not among them.

4.4 Confidence

The verification pipeline reports a confidence value and a label with every result. On the test split it reported the same confidence, 0.973 with the label HIGH, for every supported and every unsupported statement; as a score for separating them its area under the ROC curve is 0.5, and read as a probability its expected calibration error is 0.63 [MEASURED] [32, 33]. For a constant confidence that error is simply the distance between the constant and the share of supported items, here 0.973 against 1,396 of 4,053, so it describes the benchmark's mixture rather than any deployment [ANALYTICAL]. A confidence value that does not depend on the evidence measures nothing, and shown to a user as HIGH it is a false signal. A confidence meant to be read as a probability must be validated as one, against outcomes and with a proper scoring rule such as the Brier score [34]; no Lawie component produces a calibrated probability of correctness [ABSENT].

4.5 Quotations

Quotation is the one place where a verifier can be exact, and LLRB-1 tests three ways of checking one.

Lawie's quoted-holding matcher. Given the provision text, its most favourable input, it accepted 1,956 of 1,959 decisively edited quotations and 51.6% of misattributed ones; given the title and metadata it receives in deployment, it accepted 1.0% of authentic quotations [MEASURED]. With the text it is all but blind; without it, it is useless.

Exact-span authority. It accepted every authentic quotation and rejected every typographic variant and every edit; with the span derivation that EU texts would need, which is not admitted, it rejected everything [MEASURED]. It is sound and brittle: it never passed an altered quotation, and it never passed a correct one that differed by a no-break space.

Canonical matching. A reference baseline that matches substrings after folding Unicode compatibility forms, case, quotation marks, dashes and spaces accepted every authentic quotation and typographic variant, and rejected every edit and misattribution [MEASURED].

Proposition 4.2 (Exact and canonical matching)**.** *Let κ map every character of a set of folded classes (compatibility forms, case, quotation marks, dashes, spaces) to a representative. (a) A byte-exact check accepts φ against s if and only if φ is a byte substring of* $\mathsf{text}(s)$*; it therefore rejects every quotation that differs from the text in any byte, typographic variants included. (b) A canonical check accepts φ if and only if $\kappa(\varphi)$ is a substring of $\kappa(\mathsf{text}(s))$; it therefore accepts every variant that differs from an exact quotation only inside the folded classes, and accepts an edited quotation only if the edited text, canonicalised, itself occurs in the provision.*

Proof. Both parts restate the definitions: equality of bytes in (a), equality of canonical forms in (b). A variant inside the folded classes has the same canonical form as the exact quotation. □

Proposition 4.2 is the register's analytical claim [ANALYTICAL], and it marks the limit of the result. LLRB-1's supported items are quotations and typographic variants, so canonical matching scores perfectly here and would reject every legitimate paraphrase in use. Folding also has a cost of its own: compatibility normalisation maps a superscript digit to an ordinary one ("10²" and "102" share a canonical form) and case folding merges a capitalised defined term with the common word, so a canonical match cannot separate statements that differ only in folded characters [ANALYTICAL]. Exactness for quotations says nothing about paraphrase, which is where most legal writing lives.

4.6 Citations

A citation must be read before it can be verified. Lawie's citation extractor resolved the instrument for none of the 114 citations written in the official style "Regulation (EU) 2016/679", for all 49 written without "(EU)" and for 37.7% of short-name citations, and it found no article in citations abbreviated "Art." or written in Czech [MEASURED]. The official style is the style of the Official Journal itself (table 5).

Table 5: Lawie's citation extractor by citation format, test split.

Citation format (example)	n	Article found (%)	Instrument resolved (%)
Article 5(1) of Regulation (EU) 2016/679	58	100	0
Article 5(1), point (a), of Regulation (EU) …	56	100	0
Article 5(1) of Regulation 2016/679	49	100	100
Article 5(1) GDPR	53	100	38
Art. 5(1) GDPR	53	0	0
čl. 5 odst. 1 nařízení (EU) 2016/679	56	0	0

4.7 Release discipline

Three further families test the rules that decide whether an answer leaves the system.

Citation required. The gateway's rule blocked every legal conclusion that carried no citation (4 of 4) and none that carried a fabricated one (0 of 8) [MEASURED]. Every fabricated citation passed the rule, which is why the doctrine of this suite is stated as *no anchored source, no decisive claim*.

Jurisdiction. In answers to a Czech matter the gateway flagged foreign-law citations only for United States law (8 of 8); Slovak, German, Austrian, United Kingdom and Polish citations were never flagged (0 of 32) [MEASURED]. Treating Czech, Slovak and EU law as one bloc is a design choice; its effect is that a Czech answer resting on Slovak law passes as domestic.

Degraded review. Under the degraded-review policy, the default outside production, the pipeline released, with HIGH confidence, a claim that had no retrieved source but carried a citation string; under fail-closed it blocked all four tested configurations [MEASURED]. A permissive default outside production means that what is demonstrated outside production is not what production would do.

4.8 What the measurements show

1. Lawie's grounding check measures resemblance, and resemblance is not support. By corollary 2.1, it cannot reduce the decisive-error rate of what it releases on errors like LLRB-1's edits [MEASURED].
2. The components that behaved exactly are the narrow ones: exact-span authority for quotations [MEASURED], and the typed prototype for the element types it types [MEASURED]. Exactness was bought with brittleness in the first case and with blindness to unmodelled types in the second [MEASURED].
3. Reported confidence carried no information [MEASURED].
4. Citation reading fails on the Official Journal's own citation style [MEASURED], and the citation-required rule passes fabrications [MEASURED].

None of these results measures a running service or a generated answer; the study of generated answers is defined and not run [PROPOSED]. The results measure what the components decide for inputs whose correct label is known, which is a precondition for any statement about released error and not a substitute for it.

5 Legal time

A legal proposition is true or false as of a date, and it is known as of a date. A system that stores law as text and answers as if the present were the only date fails in two ways: it applies provisions before they apply, and it answers with the law as it stood when the system learned it. This section separates the dates, shows how the AI Act's application schedule was rewritten in July 2026, and measures what the usual simplifications cost. LL-TMP-01 develops the temporal model in full.

5.1 Four dates and two times

Definition 5.1 (Legal dates)**.** A provision s has a date of publication $t_{\text{pub}}(s)$, a date of entry into force $t_{\text{eif}}(s)$, a date of application $t_{\text{app}}(s)$ and a date of repeal or end of validity $t_{\text{rep}}(s)$; usually $t_{\text{pub}} \le t_{\text{eif}} \le t_{\text{app}} < t_{\text{rep}}$. The dates belong to the provision, not to the act: one act can give different provisions different dates of application, make application depend on facts other than a date, and later amend any of them. A date-determined provision is *applicable* at t_q, written $\mathsf{App}(s, t_q)$, if and only if $t_{\text{app}}(s) \le t_q < t_{\text{rep}}(s)$.

Two times index every legal question. The as-of date t_q is the date the question is about; the knowledge date t_k is the date at which the law is known. An answer given at t_k about t_q must use the schedule as amended by every act published by t_k. In the vocabulary of temporal databases t_q is valid time and t_k transaction time, and a legal corpus that is to answer historical and current questions correctly must keep both [35]. Legal informatics distinguishes the time during which a norm is in force from the times at which it is efficacious and applied [36], legislative mark-up standards record the events of a provision's life cycle [37], and half-open intervals avoid the boundary ambiguities that closed intervals create between consecutive versions [38].

Every act of the LLRB-1 corpus separates entry into force from application (table 6). The schedules encoded from the acts' final provisions as published in the Official Journal list, for the GDPR, entry into force on 24 May 2016 and application from 25 May 2018; for DORA, 16 January 2023 and 17 January 2025; for MiCA, 29 June 2023 and 30 December 2024, with its Titles III and IV from 30 June 2024; for the Digital Services Act, 16 November 2022 and 17 February 2024; for the Data Act, 11 January 2024 and 12 September 2025; for the Cyber Resilience Act, 10 December 2024 and 11 December 2027 [EXTERNAL]. Every applicability date in the benchmark's schedules carries a verbatim quotation of the act's final provisions, and all 26 quotations were found in the official text [MEASURED].

Table 6: The eight acts of the LLRB-1 corpus with their dates and their article and provision counts. "Applies (general)" is the general date of application as encoded from the act's final provisions; several acts set other dates for particular provisions.

Act	CELEX	Published	In force	Applies (general)	Articles	Provisions
GDPR	32016R0679	2016-05-04	2016-05-24	2018-05-25	99	389
DORA	32022R2554	2022-12-27	2023-01-16	2025-01-17	64	267
NIS2	32022L2555	2022-12-27	2023-01-16	2024-10-18	46	199
MiCA	32023R1114	2023-06-09	2023-06-29	2024-12-30	149	751
DSA	32022R2065	2022-10-27	2022-11-16	2024-02-17	93	363
Data Act	32023R2854	2023-12-22	2024-01-11	2025-09-12	50	235
CRA	32024R2847	2024-11-20	2024-12-10	2027-12-11	71	305
AI Act	32024R1689	2024-07-12	2024-08-01	2026-08-02	113	519

Proposition 5.1 (In force is not applicable)**.** *An index that marks a provision as current from its entry into force answers "applicable" wrongly on every date in* $[t_{\text{eif}}, t_{\text{app}})$*. An index that marks it as current from its publication answers "in force" wrongly on every date in* $[t_{\text{pub}}, t_{\text{eif}})$ *and "applicable" wrongly on every date in* $[t_{\text{pub}}, t_{\text{app}})$*.*

Proof. On a date $t_q \in [t_{\text{eif}}, t_{\text{app}})$ the first index returns the provision as current, while $\mathsf{App}(s, t_q)$ is false by definition 5.1. The second case is the same argument with t_{pub} in place of t_{eif}. □

The proposition is the register's analytical claim [ANALYTICAL]. Its force lies in the length of the intervals: for the general dates of the acts above, $[t_{\text{eif}}, t_{\text{app}})$ lasts between about fifteen months and three years.

5.2 The AI Act's Article 113, rewritten

As published in 2024, Article 113 of the AI Act set its general date of application at 2 August 2026, with Chapters I and II from 2 February 2025, the governance, general-purpose model and penalty provisions from 2 August 2025 with the exception of Article 101 (which took the general date), and Article 6(1) with its corresponding obligations from 2 August 2027 [EXTERNAL] [39]. The original point (c) of Article 113 reads:

> "(c) Article 6(1) and the corresponding obligations in this Regulation shall apply from 2 August 2027."

Regulation (EU) 2026/1744 of 8 July 2026, which under its final provisions entered into force on 27 July 2026, amended Article 113: in its text as published in the Official Journal, the application of Chapter III Sections 1–3, except Article 6(5), moves to 2 December 2027 for high-risk systems under Annex III and to 2 August 2028 for those under Annex I, and it sets 27 July 2026 for Articles 102–110 and 2 December 2026 for new prohibitions in Article 5 [EXTERNAL] [40]. The consolidated text of 27 July 2026 dates the high-risk regime by system category [41]:

> "(i) 2 December 2027 as regards AI systems classified as high-risk pursuant to Article 6(2) and Annex III; and (ii) 2 August 2028 as regards AI systems classified as high-risk pursuant to Article 6(1) and Annex I;"

EUR-Lex lists 88 modifications of the AI Act by the amending regulation, 83 of them at article level [EXTERNAL]. Table 7 sets the two schedules side by side.

Table 7: Article 113 of the AI Act as published and as amended by Regulation (EU) 2026/1744. *The scope of "the corresponding obligations" in the original point (c) is a matter of interpretation; LLRB-1 excludes the Annex I items it would date.

Provisions	As published (2024)	As amended by Reg. (EU) 2026/1744
Chapters I and II (definitions, prohibited practices)	2 February 2025	2 February 2025; new Art. 5 prohibitions 2 December 2026
Chapter III Section 4, Chapters V, VII, XII (except Art. 101), Art. 78	2 August 2025	2 August 2025
Articles 102–110 (amendments to other acts)	2 August 2026	27 July 2026
Chapter III Sections 1–3 (except Art. 6(5)), Annex III high-risk systems	2 August 2026	2 December 2027
Chapter III Sections 1–3 (except Art. 6(5)), Annex I high-risk systems	2 August 2027*	2 August 2028
All other provisions	2 August 2026	2 August 2026

The amendment is a natural experiment in legal time. Until 27 July 2026 a system that followed the published text of Article 113 answered that the obligations for Annex III high-risk systems would apply from 2 August 2026; the amended text lists 2 December 2027. A system whose schedule was fixed before the amendment disagrees with the amended text about those provisions on every day from 2 August 2026 to 1 December 2027; a system keyed on entry into force has treated them as applicable since 1 August 2024. Both errors are invisible to a user who sees only a confident date.

5.3 What the simplifications cost

LLRB-1 counts, for every provision of the corpus and every day from 2016 to 2028, whether a temporal heuristic agrees with the schedule encoded from the official text. Treating entry into force as applicability contradicts the schedules on 7–20% of provision-days, depending on the act; ignoring provision-specific dates contradicts them on up to 6%; and an AI Act schedule frozen before Regulation (EU) 2026/1744 errs on 2.0% of Annex III provision-days [MEASURED] (table 8). The 2.0% dilutes the failure over thirteen years and every provision: for each provision the amendment moved, the frozen schedule disagrees with the amended one on every day from 2 August 2026 to 1 December 2027. These are not error rates of answers. They measure how much of the legal calendar each heuristic gets wrong, and they are large because the encoded schedules of these acts defer general application by fifteen months to three years after entry into force.

Table 8: Share of provision-days, 2016–2028, on which a temporal heuristic contradicts the schedules encoded from the official texts (%), by act. Heuristics: validity from publication; validity from entry into force; the general application date for every provision; the AI Act schedule as it stood before Regulation (EU) 2026/1744. A dash marks a heuristic that does not apply to the act, or whose count for it is only partial.

Act	From publication	From entry into force	General date only	Frozen before 2026/1744
AI Act	12.4	12.0	3.4	–
AI Act (Annex I systems)	16.1	15.7	5.7	–
AI Act (Annex III systems)	15.1	14.7	4.7	2.0
CRA	20.9	20.5	2.6	–
DORA	15.8	15.4	0.0	–
DSA	7.3	6.9	2.7	–
Data Act	13.3	12.8	0.0	–
GDPR	15.8	15.4	0.0	–
MiCA	9.9	9.4	2.1	–
NIS2	13.9	13.5	0.0	–

5.4 What Lawie's temporal components do

Lawie's temporal index returns the version of a provision in force at a date over half-open intervals [TESTED]. The EUR-Lex parser starts each version at the entry-into-force date (on its metadata path, at the effective date or else the publication date) and leaves its end open [IMPLEMENTED]; dates of application, transitional provisions and consolidated versions are not modelled [ABSENT]. The index has two answers, in force or not; LLRB-1 has three states. On every test date on which a provision was in force but not yet applicable, the index and the authority predicate's in-force check reported it as in force, 781 times out of 781, and they were correct on every other date; a variant that starts versions at publication also erred on 791 of the 1,298 dates before entry into force [MEASURED] (table 9). This is proposition 5.1 observed: the components answer the question they were built for, and it is not the question a legal user asks.

Table 9: Errors of Lawie's temporal components by the true state of the provision on the test date. Index (entry into force) and index (publication) are the temporal index populated as the parser's two paths populate it.

True state on the date	n	Index (entry into force)	Authority in-force check	Index (publication)
Not yet in force	1,298	0 (0%)	0 (0%)	791 (61%)
In force, not yet applicable	781	781 (100%)	781 (100%)	781 (100%)
Applicable	456	0 (0%)	0 (0%)	0 (0%)

5.5 Detecting the change itself

A system that keeps its schedules current must first notice that the law has moved. Run on the AI Act as published and as consolidated on 27 July 2026, Lawie's legislative-diff engine found 82 of the 83 article-level amendments made by Regulation (EU) 2026/1744. On the text as EUR-Lex renders it, 21.6% of its 477 flagged changes were amendments; on canonicalised text, 86.8% of 114. It classified the postponement of application dates as a new obligation [MEASURED] (table 10). Canonicalisation removed most of the false changes, so most of them were differences of rendering, not of law. The misclassification is the temporal failure of the previous subsection in another form: an engine with no concept of a date of application can describe a moved date only as a changed duty. LL-RI-01 reports the study.

Table 10: Lawie's legislative-diff engine on the AI Act amendment of 2026. Recall is over the 83 article-level modifications listed by EUR-Lex; precision is over the changes the engine flagged.

Input to the diff engine	Changes flagged	Precision % (95% CI)	Recall % (95% CI)
Text as rendered by EUR-Lex	477	21.6 (18.1–25.5)	98.8 (93.5–99.8)
Canonicalised text	114	86.8 (79.4–91.9)	98.8 (93.5–99.8)

6 Authority: approvals, tenants and what an evaluator must see

The second path of authority is the right to act. Definition 2.8 states it as five conjuncts: binding, validity, separation, role and quorum, with delegation bounded by the parent's authority. This section describes the controls Lawie's code implements, what BEN-02 measured in aggregate, and what any authorization layer must receive to enforce each conjunct. LL-BEN-02 reports the benchmark design in full; LL-GMA-01 treats governed multi-agent execution and LL-SEC-01 security and accountability.

6.1 What the code implements

Section 3 described Lawie's approval, grant and tenant controls. BEN-02 exercises some of them directly and treats the others as context.

6.2 What BEN-02 measured

BEN-02 states seven norms that an execution request must meet: binding; validity; separation of duties; a quorum of distinct reviewers; tenant consistency of the request and of every tenant reference in the tool input, under any field name; delegation within the parent's authority; and refusal, or routing to a human, of requests induced by untrusted content. It labels 552 synthetic scenarios in sixteen attack classes and four control classes by those norms, never by the behaviour of a system under test. The functions evaluated are Lawie's tenant scoping of the request, the tenant check of the tool input and the approval evaluator, imported unchanged and run offline [MEASURED]; the database-backed grant decision and the runtime policy engine are not exercised. Three reference systems frame the result: allow everything, allow every listed tool, and allow whenever an approved gate exists for the tool, whatever its version, input, reviewer or expiry (table 11).

Across the 552 scenarios Lawie's evaluated functions denied none of the 96 legitimate requests, refused every scenario of some attack classes and allowed every scenario of others; the class-level outcomes are recorded in restricted engineering records pending remediation [MEASURED].

The reference systems are instructive in themselves. The unbound-approval system represents a design in which an approval is a flag on a tool; it refused only rejected and pending approvals. Everything that

Table 11: BEN-02 scenario classes and the reference systems: scenarios allowed out of n. †Structural classes: whether a function can refuse them depends on whether its interface receives the request's provenance or the parent's restrictions (lemma 2.1). The table reports reference systems only.

Scenario class	Norm	n	Permissive	Allow-list	Unbound approval
Valid, exactly bound approval	allow	24	24	24	24
Tool without approval requirement	allow	24	24	24	24
Own tenant named explicitly	allow	24	24	24	24
Quorum by two distinct reviewers	allow	24	24	24	24
A1 approval reused for another input	deny	24	24	24	24
A2 approval for another tool version	deny	24	24	24	24
A3 expired approval	deny	24	24	24	24
A4 reviewed before requested	deny	24	24	24	24
A4 no named reviewer	deny	24	24	24	24
A5 rejected approval	deny	24	24	24	0
A5 pending approval	deny	24	24	24	0
A6 self-approval	deny	24	24	24	24
A7 two approvals by one person	deny	24	24	24	24
A7b quorum with named reviewers, one person	deny	24	24	24	24
A8 foreign tenant requested	deny	24	24	24	24
A9 foreign tenant in a conventional tenant field	deny	24	24	24	24
A10 foreign tenant under an alternative field name, top level	deny	96	96	96	96
A10 foreign tenant under an alternative field name, nested	deny	24	24	24	24
A11 request induced by untrusted content†	deny	24	24	24	24
A12 sub-agent beyond parent's restrictions†	deny	24	24	24	24

binding to version, input hash, reviewer and time refuses, the flag design allows. This is Position 4 measured on a reference system: an approval that is not bound to content authorizes whatever content arrives.

6.3 What an evaluator must receive

Lemma 2.1 turns each conjunct of the model into a requirement on the interface of any authorization layer, whatever its implementation.

Binding. To refuse a substituted input or a drifted version, the evaluator must recompute the hash of the canonical current input at the moment of use and compare tool, version and hash with the approval.

Validity. To refuse stale, premature or manufactured consent, it must receive the decision, the reviewer and the times of request, review, use and expiry.

Separation of duties. To refuse self-approval, the evaluator must receive the requester's identity with every approval and compare it with the reviewer's.

Quorum. To enforce n distinct reviewers, the evaluator must count distinct persons, not approvals.

Provenance. To refuse a request induced by untrusted content, such as an instruction inside a retrieved document or a counterparty's e-mail, the function that decides must learn the request's provenance. This is the confused deputy of Hardy [42]: an agent holds the principal's authority and takes instructions from a party that holds none. Prompt injection [43] in its indirect form [44] is the present version of it; agent benchmarks show it succeeding against tool-using agents [45, 46], and it heads the OWASP list of risks for LLM applications [47].

Delegation. To keep $C(a') \subseteq C(a)$, a child's authorization must receive its parent's restrictions and approval requirements.

Tenant references in tool input. A check that enumerates where a tenant identifier may appear fails open on every place it did not enumerate. The norm in BEN-02 therefore says "under any field name", and the benchmark varies field names and nesting to test it.

6.4 What it means

Separation of privilege, a mechanism that needs two keys being harder to subvert than one that needs one [13], and separation of duty in commercial integrity policies [12] both presuppose that the system knows who holds each key. A predicate cannot keep apart persons it cannot name, and it cannot refuse instructions whose origin it never sees. A benchmark of authorization must therefore include classes whose outcome is fixed by the interface, and report them separately from the classes an implementation can fix by changing its logic.

A refusal in BEN-02 shows that the evaluated function refused. Whether every route in the code calls the function would have to be established from source; the benchmark does not establish it, and the norms are Lawie Labs' statement of what authorization requires; another institution may adopt different ones (section 9).

7 Evidence: chains, anchors and dossiers

Evidence is what an act leaves for the hostile reader. Lawie records two kinds: hash-chained audit records of agent runs, and signed portable dossiers of institutional execution. This section states what each can show and what the evidence-integrity study (EVD) measured. LL-EVP-01 develops the argument in full.

7.1 What a chain can show

Proposition 7.1 (Chains need witnesses)**.** *Let $\mathscr{C}$ be a chain as in definition 2.9, built with an unkeyed hash whose output has fixed length.*

(a) Every prefix of a verifying chain verifies, so truncating the tail leaves no trace in the chain.

(b) Anyone who can write the records and compute $\mathscr{H}$ can produce a verifying chain for any sequence of records, so rewriting followed by re-hashing leaves no trace in the chain.

(c) If a witness independent of the writer holds (n, h_n), every change to $(r_1, \dots, r_n)$ is detected by comparison with the witness, unless it produces a collision of $\mathscr{H}$.

Proof. (a) The checks for $h_1, \dots, h_m$ use only $r_1, \dots, r_m$. (b) Recompute every h_i from the new records. (c) A change of length changes n. Otherwise let (r'_i) differ from (r_i) with $h'_n = h_n$, and let i be the largest index with $(h'_{i-1}, r'_i) \neq (h_{i-1}, r_i)$; such an index exists because the sequences differ. Since h_{i-1} has fixed length, the concatenation $h_{i-1} \,\|\, r_i$ determines the pair, so $\mathscr{H}(h'_{i-1} \,\|\, r'_i) = h'_i = h_i = \mathscr{H}(h_{i-1} \,\|\, r_i)$ is a collision; the equality $h'_i = h_i$ holds because every later pair is unchanged and $h'_n = h_n$. □

Proposition 7.1 is the register's analytical claim [ANALYTICAL]. It is the reason time-stamping links a document to an external publication [14], why secure audit logs protect the entries written before a compromise and not those after it [48], and why transparency logs publish signed heads and consistency proofs [15, 49].

The evidence-integrity study (EVD) applied in-place mutations to Lawie's agent audit chain, 50 trials each. The chain verifier detected every tested edit, deletion, reordering and insertion of records and every deleted prefix (table 12), and the portable dossier verifier failed every tampered dossier [MEASURED].

Proposition 7.1 states what any chain-based audit trail needs to be evidence against its own keeper. By proposition 7.1(b), a party that can recompute an unkeyed chain leaves no trace in it; only a witness of

Table 12: In-place mutations of Lawie's agent audit chain detected by its verifier.

Mutation	Trials	Detected by the chain verifier
Edit a record	50	50
Delete a record	50	50
Swap two records	50	50
Insert a forged record	50	50
Delete a prefix	50	50

the head that the operator does not control, such as a time-stamping service or a public transparency log, makes truncation and rewriting detectable by a third party (proposition 7.1(c)). These are requirements of the model for any evidence system; they are not satisfied by a hash chain alone.

7.2 Signed dossiers

Lawie signs institutional dossiers with an asymmetric key held in a managed key-management service [IMPLEMENTED], and the dossiers verify offline against pinned key fingerprints [TESTED]. In the EVD study the verifier failed every tampered dossier (an edited record, a dropped record, a signature by a foreign key, a key that the trust policy did not pin) and verified the intact dossier [MEASURED].

For any signed-evidence format, a dossier relied on long after signing needs two things. The time of signing must be witnessed by an independent time-stamp authority rather than asserted by the signer [14, 50], because a time recorded by the signer is only the signer's statement. And the status of a key, its validity and its revocation, must be available from a source outside the producer's control, because a producer who reports its own key status can report it as it likes.

A dossier manifest is valid only if it states, inside the hashed and signed manifest, that verification proves neither substantive truth nor admissibility nor legal correctness, and creates no authority [IMPLEMENTED]. Dossiers cover institutional execution, not agent runs.

7.3 Integrity is not truth

Table 1 drew the line, and the dossier manifest states it inside the artifact. A hash shows that bytes have not changed since they were hashed; a signature shows that a key holder attested to them. Neither shows that what the bytes say is true, that the record is complete, or that a court would admit it. Provenance models such as W3C PROV make a declared chain of origin explicit [51]; they do not make it true. An evidence system earns reliance by stating these limits in the artifact, where the reader cannot miss them, and by making every stronger claim depend on a witness outside the operator's control.

8 Civic intelligence: accountable publication and the base rate

Civic Guard applies the same doctrine to public-interest work: official public records such as legislation, procurement notices and funding data are turned into findings that may name an institution. Here the cost of a false statement falls on a third party who never used the system, and the design is built around that asymmetry. Civic Guard is quarantined: none of its release stages is admitted and its public pages are not served in production [BLOCKED]. LL-CG-01 reports the evaluation programme in full.

8.1 Design

Findings start from source observations that record payload hashes and retrieval times [IMPLEMENTED]. Screens are deterministic rule cards whose outcomes are triggered, not triggered or abstained; they are marked not calibrated, and they are unapproved drafts [IMPLEMENTED]. Publication requires a reviewer other than the investigator and a publisher other than the reviewer, a separate authorization for legal language, and published facts whose hashes match confirmed facts [IMPLEMENTED]. A Czech right-of-reply policy gives an implicated entity a window of at least seven days after evidenced delivery, and a right-of-reply interface is implemented [IMPLEMENTED]. Severity is an editorial prioritisation label, not a calibrated probability, and uncertainty is recorded as free text [IMPLEMENTED]. The product excludes the scoring of persons and automatic public accusation, and a repository check blocks wording such as "guilty", "AI decides" and "verified evidence" in its copy [IMPLEMENTED]. Standards for open-source investigation, such as the Berkeley Protocol, ask for comparable discipline in documenting, preserving and verifying digital information [52].

The publication path separates investigator, reviewer and publisher by person. Separation of duties is not a property of a platform; it is a property of each path through it, and each path has to be designed and tested for it.

8.2 The base rate

Proposition 8.1 (Base-rate bound on screening alerts)**.** *Let a screen trigger on a share t of cases, and let true anomalies have prevalence p. Then the share of alerts that are true anomalies, the screen's positive predictive value, satisfies* $\mathrm{PPV} \le p/t$*, with equality only if the screen catches every anomaly.*

Proof. With A the event that a case is anomalous and T that the screen triggers, $\mathrm{PPV} = \Pr[A \mid T] = \Pr[T \mid A]\ \Pr[A]/\Pr[T] = \Pr[T \mid A]\, p/t \le p/t$, with equality if and only if $\Pr[T \mid A] = 1$. □

The bound is the register's analytical claim [ANALYTICAL]. In 2024, 40% of Czech public contract awards had a single bidder, against an EU average of 28% [EXTERNAL] [53], and single bidding is a standard corruption-risk indicator [54]. A screen that flagged every single-bid award would therefore trigger on 40% of Czech awards, and whatever the prevalence p of irregular procedures, at most $p/0.40$ of its alerts would concern one. If one award in ten were irregular, a figure chosen for illustration, at least three alerts in four would concern a procedure that was not, and unless more than one award in five were irregular, most alerts would concern regular procedures. A trigger is a reason to look, never a finding. The publication controls above exist because the arithmetic makes most triggers innocent wherever the flagged pattern is much more common than the irregularity it indicates.

8.3 What was measured and what was not

No detection accuracy has been measured. No sealed evaluation corpus exists, and precision, recall and false-positive rates have never been measured [ABSENT]; the twelve official sources of the production profile are not admitted [BLOCKED]; and the studies of alert burden, false positives, calibration, publication accountability and correction propagation are defined and wait on external owners [BLOCKED].

9 Limitations

The limitations below bound every result in this paper. Some are properties of the method, some of the evidence that exists, and some of the authors' position. A reader should weigh them as heavily as the findings.

9.1 What LLRB-1 does not measure

1. *Generated answers.* LLRB-1 measures verification components, not language models. The study of generated answers, which would measure fabricated citations, misquotation, temporal error and abstention in model output and the change in released error when a verification layer gates it, is defined and not run [PROPOSED]. Without it the base rate and the type mix of decisive errors in Lawie's generated answers are unknown, and proposition 2.1 cannot be evaluated for any deployed configuration (example 2.1).
2. *Paraphrase.* Supported items are quotations and typographic variants; no item tests whether a verifier accepts a correct paraphrase. A verifier that accepts only quotations scores perfectly here and would reject legitimate paraphrase in use.
3. *Error coverage.* Each edited item carries exactly one edit, from nine families. Real errors combine, and the families are not a census of the ways a legal statement can be wrong. Corollary 2.2 applies to the benchmark itself: its results say nothing about error types it does not contain.
4. *Law covered.* The corpus is English-language EU legislation. National law, case law and contracts are not covered, although the only span derivation Lawie admits is for Czech statutes [IMPLEMENTED].
5. *Isolation.* Components ran in isolation, with inputs that production would read from storage constructed from the official text, not through a running service. Results under other inputs, such as several retrieved texts per claim, are not measured.
6. *Labels and hypotheses.* Labels are fixed by construction and no lawyer has reviewed them [ABSENT], so no agreement statistic between independent annotators exists [55]. The hypotheses were chosen after exploring the development split [MEASURED]; the test split protects against tuning, not against the choice of question.

9.2 Authorization and evidence studies

BEN-02's scenarios are synthetic and exercise components, not workflows. A refusal shows that the evaluated function refused; whether every route in the code calls it would have to be established from source; the benchmark does not establish it. The database-backed grant decision and the runtime policy engine were not exercised, and the norms are Lawie Labs' own. EVD mutated synthetic chains and dossier fixtures; it says nothing about the operational custody of keys or databases.

9.3 Civic Guard

No detection accuracy of Civic Guard has been measured [ABSENT]. The base-rate bound needs the prevalence of irregular procedures, which this paper does not know; it bounds the value of an alert, not the quality of any screen.

9.4 The analytical results

Proposition 2.1 and its corollaries assume that F and R are stable over the population a gate serves; in practice both shift with the distribution (definition 2.6). Distribution-free coverage bounds, such as those of conformal prediction, require calibration data exchangeable with the data of use [56]; no Lawie component has such a calibration. Proposition 2.2(b) and (c) assume independence between steps. Proposition 7.1 assumes a collision-resistant hash. Proposition 8.1 assumes that prevalence and trigger share refer to the same population of cases.

9.5 Independence and review

The organisation that built the components designed and ran the benchmarks. Reporting standards for machine-learning-based science ask for the disclosures that make such work checkable, from data provenance to the risk of leakage between development and test data [57]. Pre-registration, labels fixed by construction, recorded component hashes and published negative results narrow the room for self-serving analysis; they do not replace independent replication, which has not happened. No independent audit, certification or expert review of the components exists [ABSENT], and this paper has not been peer reviewed. Lawie's evaluation laboratory has never run a model: its live evaluation path throws, its recorded runs are fixtures, and its legal datasets named after public benchmarks are templated fixtures, not those benchmarks [ABSENT]. This programme does not change that; it measures components, not models.

9.6 Law and scope

Legal texts and dates are those of EUR-Lex as retrieved on 26 November 2026 [MEASURED]; only the Official Journal is authentic. The schedules of application used in this paper, those of Article 113 of the AI Act included, are research labels derived from the official texts; no qualified lawyer has checked them [ABSENT], and they are not legal advice. This paper reports research measurements at a stated code basis. It is not a security assurance, a compliance statement or legal advice, and no independent security, privacy or legal review of it has taken place. No statement in it says that any use of Lawie complies with any law.

10 Where it stands

Table 13: The evidence boundary at the basis commit, by area. Each entry is a statement about Lawie with the class the claim register assigns to it; the right-hand column states what is missing.

Area	In source	Measured offline	Gaps
Sources and corpus	Admission under approved source contracts; observations with hashes, licence and supersession [IMPLEMENTED]	An eight-act EU corpus built for the benchmark [MEASURED]	No production source admitted [BLOCKED]; only a Czech span derivation admitted [IMPLEMENTED]
Verification	Exact-span authority predicate [TESTED]; withholding of unbacked citations [TESTED]; citation gate [TESTED]	Grounding accepts every decisive edit [MEASURED]; confidence carries no information [MEASURED]; quotation checks blind or brittle [MEASURED]	No calibrated probability of correctness [ABSENT]; good-law checks cannot pass [IMPLEMENTED]; generated answers not yet studied [PROPOSED]
Legal time	Version intervals [TESTED]; versions start at entry into force [IMPLEMENTED]	In force read as applicable [MEASURED]; schedule exposure [MEASURED]; amendment detection [MEASURED]	Dates of application, transitions and consolidated versions not modelled [ABSENT]
Authority	Approval, grant and tenant controls (section 3)	No legitimate request denied; some attack classes refused throughout, others allowed [MEASURED]	Unresolved findings recorded in restricted engineering records pending remediation
Evidence	Hash-chained audit records; signed portable dossiers [TESTED]	In-place tampering detected; tampered dossiers fail [MEASURED]	No independent audit [ABSENT]
Civic Guard	Accountable-publication controls [IMPLEMENTED]; deterministic screens [IMPLEMENTED]	—	Quarantined [BLOCKED]; no evaluation corpus [ABSENT]; five studies wait on external owners [BLOCKED]
Product and assurance	Claims policy with truth states [IMPLEMENTED]	—	No production acceptance [ABSENT]; not deployed [BLOCKED]; no audit or expert review [ABSENT]; the evaluation laboratory has never run a model [ABSENT]; labels not reviewed by a lawyer [ABSENT]

Table 13 states the evidence boundary area by area. The middle column reports components executed offline on public legal text or synthetic scenarios; nothing in it is a production or customer measurement. The column of gaps is as much a part of the record as the other two.

The state in one paragraph. Lawie is a source repository whose verification, authorization and evidence components are implemented and partly tested; whose measured behaviour, at one commit and offline, combines exact refusals with measured failures; whose official sources are not admitted for production; whose civic product is quarantined; whose assurance is its own; and of which nothing is deployed in production. The claims of this paper are bounded by that sentence.

11 Conclusion

The argument of this paper has three steps. Legal work produces acts, and acts need authority: a warrant from an official source applicable on the relevant date, and a right to act conferred on a named person. Intelligence cannot supply either, because authority depends on facts that no model output contains (corollary 2.3). A system that executes legal work must therefore make authority executable, by anchoring, binding, separating and witnessing, and must refuse where it cannot.

Measured against that standard, Lawie's source code is uneven in a way that is itself informative. Where a requirement reduces to exact comparison, as with a span hash, an approval bound to an input hash or a signed dossier, the components behaved exactly in the tests and offline measurements reported here. Where a requirement needs judgement about meaning or time, as with support and applicability, they approximate it, and the measurements show the cost: every decisive edit accepted, every in-force-but-not-applicable date misread. In authorization the evaluated functions denied no legitimate request and refused some attack classes throughout and others not at all [MEASURED].

What a legal-execution system needs follows from the model, not from a more capable model. Check the decisive elements and name the types that are checked. Key legal time to application and to the date the law was known, not to entry into force. Make evidence verifiable by a party outside the operator's control. These requirements hold for any system that claims to execute legal work.

Nothing described in this paper is shipped. The measurements and the negative results are reported here, with claim identifiers that the internal register resolves, so that the next version of Lawie can be measured against this one by readers who have no reason to trust it.

A Evidence classes

Every material statement about Lawie in this paper carries one of the labels in table 14. The labels are not typed by hand. Each is printed from the class that the suite's claim register assigns to the claim the statement cites, and a citation of a claim that is not in the register stops the build, so a statement cannot carry a stronger label than the register allows. The classes refine the truth states of the company's claims policy [IMPLEMENTED]; none of them means deployed. The register is internal: it records the sources of each claim, which are repository paths and are not reproduced in this edition.

Table 14: Evidence classes used in this paper.

Label	Meaning
[IMPLEMENTED]	In the source repository at the basis commit.
[TESTED]	Exercised by a named repository test at the basis commit.
[MEASURED]	Measured by an offline experiment of this suite on stated public or synthetic data; never a production or customer measurement.
[ANALYTICAL]	Follows by argument from stated assumptions; derived in this paper.
[EXTERNAL]	Checked against a cited primary source, such as the Official Journal or a Commission statistical release.
[PROPOSED]	A design, model or protocol that is not built or not run.
[ABSENT]	Not implemented, or not done.
[BLOCKED]	Awaits an external owner, approval or decision.

The classes [SIMULATED] (output of a stated model) and [UNVALIDATED] (a hypothesis stated as such) are part of the suite's vocabulary and are not used in this paper.

B Index of the claims cited

Table 15 lists every register claim this paper cites, with its class, its subject and the sections that cite it. The identifiers are those of the suite's claim register, shared by all fourteen publications.

Table 15: Claims cited in this paper.

Claim	Class	Subject	Cited in
C-001	[ABSENT]	Product acceptance	Abstract, Position, §10, §11
C-002	[BLOCKED]	Production deployment	Abstract, Position, §10
C-003	[BLOCKED]	Civic Guard quarantine	Abstract, Position, §8, §10
C-004	[ABSENT]	Audit and expert review	Abstract, Position, §9, §10
C-005	[IMPLEMENTED]	Claims policy and truth states	§10, App. A
C-006	[IMPLEMENTED]	Mission	Position
C-007	[PROPOSED]	Authority model (design)	§2
C-010	[TESTED]	Citation gate on agent outputs	§3, §10
C-011	[TESTED]	Citation-authority predicate	Position, §2, §3, §10
C-012	[IMPLEMENTED]	Admitted span derivations	§3, §9, §10
C-013	[TESTED]	Withholding of unbacked citations	§3, §10
C-015	[IMPLEMENTED]	Degraded-review default	§3, §4
C-016	[IMPLEMENTED]	Good-law checks	§3, §10
C-017	[TESTED]	Temporal index intervals	§3, §5, §10
C-018	[IMPLEMENTED]	Start of EUR-Lex versions	§3, §5, §10
C-019	[ABSENT]	Application dates and transitions	§3, §5, §10
C-020	[TESTED]	Jurisdiction routing	§3
C-021	[IMPLEMENTED]	Gateway jurisdiction rule	§3, §4
C-023	[IMPLEMENTED]	Retrieval pipeline	§3
C-024	[TESTED]	Missing-document detection	§3
C-025	[ABSENT]	Confidence score	§3, §4, §9, §10
C-026	[ABSENT]	Evaluation laboratory	§9, §10
C-030	[TESTED]	Capability modes	§3, §6
C-031	[TESTED]	Tenant tool grants	§3, §6, §10
C-032	[TESTED]	Approval binding	§3, §6, §10, §11
C-033	[TESTED]	Approval validity	§3, §6, §10
C-037	[TESTED]	Orchestrator identity assertions	§3, §6
C-038	[TESTED]	Tenant scoping	§3, §6, §10
C-040	[TESTED]	Agent types	§3
C-052	[TESTED]	Signed portable dossiers	§3, §7, §10
C-053	[IMPLEMENTED]	Dossier limitation statement	§7
C-060	[IMPLEMENTED]	Civic Guard screens	§8, §10
C-061	[IMPLEMENTED]	Civic Guard publication roles	§8, §10
C-062	[IMPLEMENTED]	Right of reply	§8
C-063	[IMPLEMENTED]	Severity and uncertainty	§8
C-064	[IMPLEMENTED]	Civic Guard exclusions and wording	§8
C-065	[BLOCKED]	Production source profile	§3, §8, §10
C-066	[ABSENT]	Civic Guard evaluation corpus	§8, §9, §10
C-067	[IMPLEMENTED]	Source observations	§3, §8, §10
C-100	[MEASURED]	LLRB-1 corpus	§4, §5, §9, §10
C-101	[MEASURED]	Schedule quotations	§5
C-102	[MEASURED]	Support items and labels	§4
C-103	[MEASURED]	Pre-registration	§4
C-104	[MEASURED]	H1: grounding check	Abstract, Position, §3, §4, §10, §11
C-105	[MEASURED]	H2: calibrated lexical verifiers	Abstract, §2, §4

Claim	Class	Subject	Cited in
C-106	[MEASURED]	H3a: typed prototype, modelled edits	Abstract, §2, §4
C-107	[MEASURED]	H3b: typed prototype, held-out edits	Abstract, §2, §4
C-108	[MEASURED]	Confidence analysis	Abstract, Position, §4, §10
C-109	[MEASURED]	Quoted-holding matcher	Position, §4, §10
C-110	[MEASURED]	Exact-span authority	Position, §4, §10, §11
C-111	[MEASURED]	Canonical matching	§4
C-112	[MEASURED]	Citation extraction	§4
C-113	[MEASURED]	Temporal components	Abstract, Position, §3, §5, §10, §11
C-114	[MEASURED]	Schedule exposure	Abstract, §5, §10
C-115	[MEASURED]	Jurisdiction flagging	§3, §4
C-116	[MEASURED]	Citation-required blocking	Position, §3, §4
C-117	[MEASURED]	Degraded-review release	§4
C-118	[MEASURED]	Harness reproduction check	§4
C-124	[MEASURED]	EVD: dossiers	§7, §10, §11
C-125	[MEASURED]	Legislative diff on the AI Act amendment	§5, §10
C-200	[EXTERNAL]	AI Act Article 113 as published	§5
C-201	[EXTERNAL]	Regulation (EU) 2026/1744	§5
C-202	[EXTERNAL]	Modifications listed by EUR-Lex	§5
C-203	[EXTERNAL]	Dates of the corpus acts	§5
C-204	[EXTERNAL]	Czech single-bidder share	§8
C-300	[ANALYTICAL]	Containment and deletion	§4
C-301	[ANALYTICAL]	Exact and canonical matching	§4
C-302	[ANALYTICAL]	Base-rate bound	Position, §8
C-303	[ANALYTICAL]	Chains and witnesses	§7
C-304	[ANALYTICAL]	Released error of a gate	§2
C-305	[ANALYTICAL]	In force and applicable	§5
C-400	[PROPOSED]	Generated-answer study	Abstract, Position, §2, §4, §9, §10
C-401	[BLOCKED]	Civic Guard studies	Position, §8, §10
C-402	[PROPOSED]	Status of the typed prototype	§4
C-403	[ABSENT]	Review of the labels	Abstract, §4, §9, §10
C-500	[MEASURED]	Run provenance (files loaded)	Abstract, §4, §6
C-501	[MEASURED]	Exploration before registration	§4, §9
C-502	[TESTED]	Refusal reasons	§3, §10
C-503	[IMPLEMENTED]	Runner refusal on changed files	§4
C-557	[ANALYTICAL]	Calibration error of a constant	§4
C-562	[IMPLEMENTED]	Construction of support items	§4
C-569	[ANALYTICAL]	Costs of canonical folding	§4
C-571	[IMPLEMENTED]	Enriched family mix	§2, §4
C-605	[IMPLEMENTED]	Authenticated reviewer	§3, §6, §10
C-606	[IMPLEMENTED]	Approval request lifetime and reuse	§3
C-607	[TESTED]	Forbidden as the empty composition	§3
C-608	[TESTED]	Pause before privileged steps	§3
C-609	[TESTED]	Orchestrator ingress without keys	§3
C-613	[IMPLEMENTED]	Dossier signing key	§3, §7, §10
C-617	[IMPLEMENTED]	Grant decision reads current state	§3
C-621	[IMPLEMENTED]	Single-use approvals	§3
C-629	[MEASURED]	BEN-02 aggregate	Abstract, Position, §3, §6, §10, §11
C-655	[IMPLEMENTED]	Dossier limitation statements	§3, §7
C-659	[MEASURED]	Evidence-integrity aggregate	Abstract, §3, §7, §10

References

[1] Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why Language Models Hallucinate, 2025. arXiv:2509.04664.

[2] Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E. Ho. Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models. *Journal of Legal Analysis*, 16(1):64–93, 2024. doi: 10.1093/jla/laae003.

[3] Varun Magesh, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools. *Journal of Empirical Legal Studies*, 2025. doi: 10.1111/jels.12413.

[4] United States District Court for the Southern District of New York. Mata v. Avianca, Inc., No. 22-cv-1461 (PKC), Opinion and Order on Sanctions, 2023. URL https://www.courtlistener.com/docket/63107798/mata-v-avianca-inc/. 22 June 2023.

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

[6] Adam Tauman Kalai and Santosh S. Vempala. Calibrated Language Models Must Hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 160–171, 2024. doi: 10.1145/3618260.3649777.

[7] H. L. A. Hart. *The Concept of Law*. Clarendon Press, Oxford, 1961.

[8] Joseph Raz. *The Authority of Law: Essays on Law and Morality*. Clarendon Press, Oxford, 1979.

[9] Lawrence Lessig. *Code and Other Laws of Cyberspace*. Basic Books, New York, 1999.

[10] C. K. Chow. On Optimum Recognition Error and Reject Tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

[11] Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 4878–4887, 2017.

[12] David D. Clark and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194, 1987. doi: 10.1109/SP.1987.10001.

[13] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939.

[14] Stuart Haber and W. Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2): 99–111, 1991. doi: 10.1007/BF00196791.

[15] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, Internet Engineering Task Force, 2013.

[16] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, et al. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 36 (Datasets and Benchmarks Track)*, 2023.

[17] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When Does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset of 53,000+ Legal Holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 159–168, 2021. doi: 10.1145/3462757.3466088.

[18] Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. LexGLUE: A Benchmark Dataset for Legal Language Understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4310–4330, 2022. doi: 10.18653/v1/2022.acl-long.297.

[19] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring Attribution in Natural Language Generation Models. *Computational Linguistics*, 49(4):777–840, 2023. doi: 10.1162/coli_a_00486.

[20] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling Large Language Models to Generate Text with Citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, 2023. doi: 10.18653/v1/2023.emnlp-main.398.

[21] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, 2023. doi: 10.18653/v1/2023.emnlp-main.741.

[22] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, 2024. doi: 10.18653/v1/2024.eacl-demo.16.

[23] Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, et al. Evaluating Models' Local Decision Boundaries via Contrast Sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, 2020. doi: 10.18653/v1/2020.findings-emnlp.117.

[24] Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. In *International Conference on Learning Representations*, 2020.

[25] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, 2020. doi: 10.18653/v1/2020.acl-main.442.

[26] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, 2019. doi: 10.18653/v1/P19-1334.

[27] Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The Preregistration Revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114.

[28] Quinn McNemar. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

[29] Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6 (2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

[30] C. J. Clopper and E. S. Pearson. The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404.

[31] Edwin B. Wilson. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

[32] James A. Hanley and Barbara J. McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1):29–36, 1982. doi: 10.1148/radiology.143.1.7063747.

[33] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, 2017.

[34] Glenn W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1): 1–3, 1950. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.

[35] Christian S. Jensen, Curtis E. Dyreson, et al. The Consensus Glossary of Temporal Database Concepts — February 1998 Version. In *Temporal Databases: Research and Practice*, volume 1399 of *Lecture Notes in Computer Science*, pages 367–405. Springer, 1998. doi: 10.1007/BFb0053710.

[36] Monica Palmirani, Guido Governatori, and Giuseppe Contissa. Modelling Temporal Legal Rules. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, pages 131–135, 2011. doi: 10.1145/2018358.2018378.

[37] OASIS. Akoma Ntoso Version 1.0. Part 1: XML Vocabulary. OASIS Standard, 2018. URL https://docs.oasis-open.org/legaldocml/akn-core/v1.0/akn-core-v1.0-part1-vocabulary.html.

[38] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11): 832–843, 1983. doi: 10.1145/182.358434.

[39] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, L, 2024/1689, 12.7.2024, 2024. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32024R1689.

[40] European Parliament and Council of the European Union. Regulation (EU) 2026/1744 of 8 July 2026 amending Regulations (EU) 2024/1689, (EU) 2018/1139 and (EU) 2023/1230 as regards the simplification of the implementation of harmonised rules on artificial intelligence (Digital Omnibus on AI). Official Journal of the European Union, L, 24.7.2026, 2026. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32026R1744.

[41] Publications Office of the European Union. Regulation (EU) 2024/1689, consolidated text of 27 July 2026. EUR-Lex, document 02024R1689-20260727, 2026. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:02024R1689-20260727.

[42] Norm Hardy. The Confused Deputy (or Why Capabilities Might Have Been Invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, 1988. doi: 10.1145/54289.871709.

[43] Fábio Perez and Ian Ribeiro. Ignore Previous Prompt: Attack Techniques for Language Models, 2022. arXiv:2211.09527.

[44] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023. doi: 10.1145/3605764.3623985.

[45] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, 2024. doi: 10.18653/v1/2024.findings-acl.624.

[46] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, 2024.

[47] OWASP Foundation. OWASP Top 10 for LLM Applications 2025, 2024. URL https://genai.owasp.org/llm-top-10/.

[48] Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999. doi: 10.1145/317087.317089.

[49] Scott A. Crosby and Dan S. Wallach. Efficient Data Structures for Tamper-Evident Logging. In *Proceedings of the 18th USENIX Security Symposium*, pages 317–334, 2009. URL https://www.usenix.org/conference/usenixsecurity09/technical-sessions/presentation/efficient-data-structures-tamper-evident.

[50] Carlisle Adams, Pat Cain, Denis Pinkas, and Robert Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161, Internet Engineering Task Force, 2001.

[51] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-dm/.

[52] United Nations Office of the High Commissioner for Human Rights and Human Rights Center, UC Berkeley School of Law. Berkeley Protocol on Digital Open Source Investigations, 2022. URL https://digitallibrary.un.org/record/3973652. HR/PUB/20/2.

[53] European Commission. Single Market and Competitiveness Scoreboard: Access to Public Procurement (reporting period 2024), 2025. URL https://single-market-scoreboard.ec.europa.eu/business-framework-conditions/public-procurement_en. Accessed 26 November 2026.

[54] Mihály Fazekas and Gábor Kocsis. Uncovering High-Level Corruption: Cross-National Objective Corruption Risk Indicators Using Public Procurement Data. *British Journal of Political Science*, 50(1):155–164, 2020. doi: 10.1017/S0007123417000461.

[55] Klaus Krippendorff. Reliability in Content Analysis: Some Common Misconceptions and Recommendations. *Human Communication Research*, 30(3):411–433, 2004. doi: 10.1111/j.1468-2958.2004.tb00738.x.

[56] Anastasios N. Angelopoulos and Stephen Bates. Conformal Prediction: A Gentle Introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. doi: 10.1561/2200000101.

[57] Sayash Kapoor, Emily M. Cantrell, Kenny Peng, Thanh Hien Pham, Christopher A. Bail, et al. REFORMS: Consensus-based Recommendations for Machine-learning-based Science. *Science Advances*, 10(18): eadk3452, 2024. doi: 10.1126/sciadv.adk3452.