The Witness Register: Co-Registration Without Aggregation

A shared register for line report envelopes that never ranks, merges, or overrides the instruments it holds

Daniel Ari Friedman

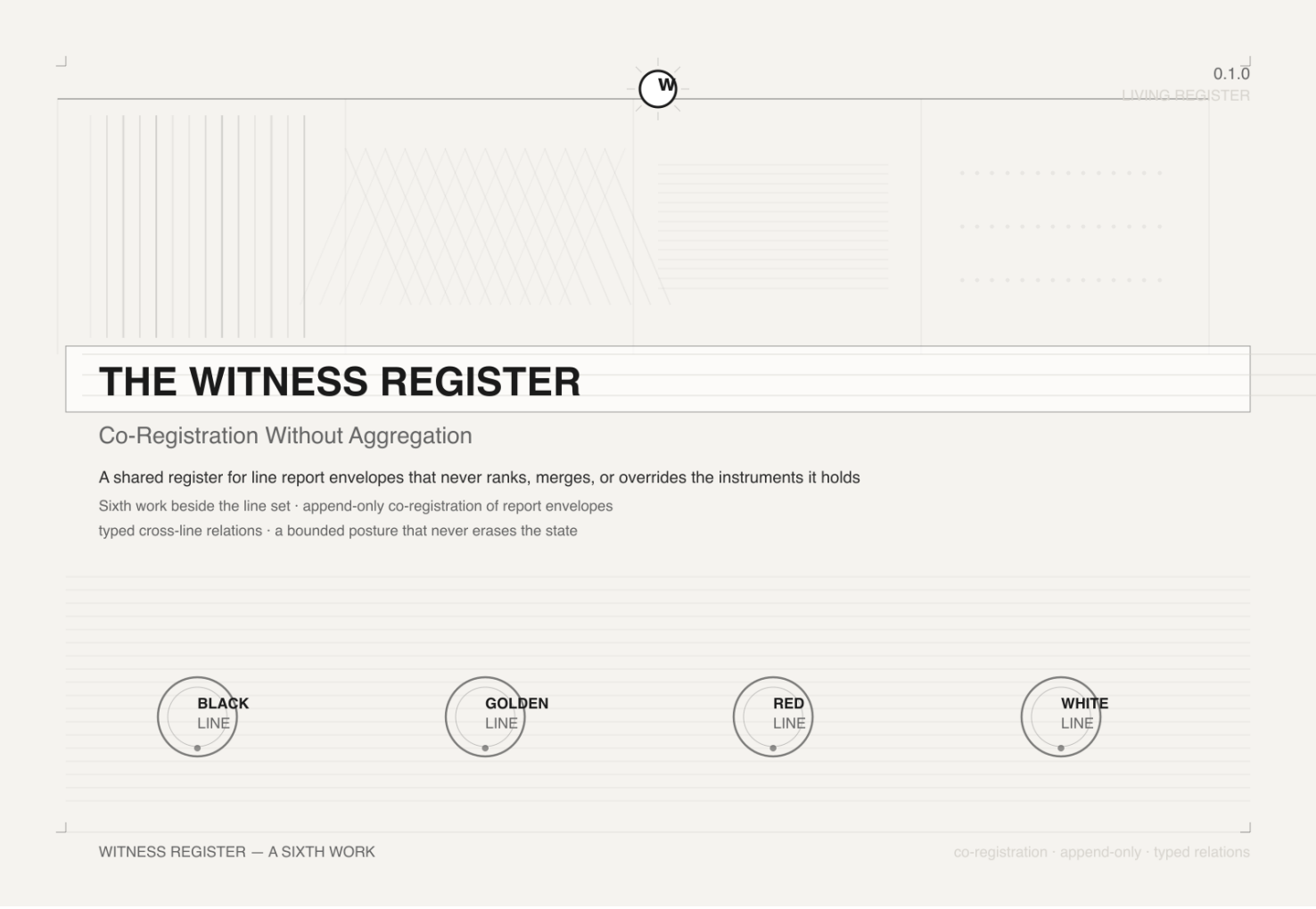
Active Inference Institute

daniel@activeinference.institute

ORCID: 0000-0001-6232-9096

DOI: 10.5281/zenodo.21754246

2026-07-29



Contents

1 Abstract

The Witness Register is a shared co-registration layer for the four-line set, built as a sixth work standing beside the instruments rather than among them. Each line already exports one common report envelope under the published schema string `line.report-envelope/1.0` — a digest pointer to its complete native report, its identity, subject, review date, registry provenance, native status in its own vocabulary, and its non-claims. The register accepts those envelopes by value, holds them beside one another without reading past their covers, and records cross-line relations as separate authored records — non-compensatory block, unresolved dependency, protected absence, directional tension, unclassified observation, return due, cannot-compare, and agreement.

The register's defining properties are refusals, enforced in code: it never imports a line package; it never parses, compares, ranks, averages, or merges any line's `native_status`; it never auto-creates a category; it never infers consent or permission; it never mutates or rewrites history; and it emits no score other than one bounded posture, and that only on request. States are sealed by a digest over their complete canonical content and chained by `prior_ref`, so in-place tampering fails closed. On request, for one declared next use, the register projects `-1 | 0 | +1` under non-compensatory invariants: an unresolved block forces `-1`; a protected absence forbids `+1` unconditionally; an empty register is `-1`, because nothing to witness is not permission. Every projection carries the digest of the state that earned it and the reasons it holds.

What the register does not do is as important as what it does. The tip is unbound without an external anchor the chain does not control. Intake is a shape check, never a truth check. Relations are authored by people, and a missing relation is invisible to projection. The posture is not a decision, not a utility, and not permission from any boundary owner. Three first-pass measures — relation fidelity, return recoverability, premature crowning rate — evaluate the register's own bookkeeping, never the truth of any report. The register witnesses; it does not judge.

2 The Problem

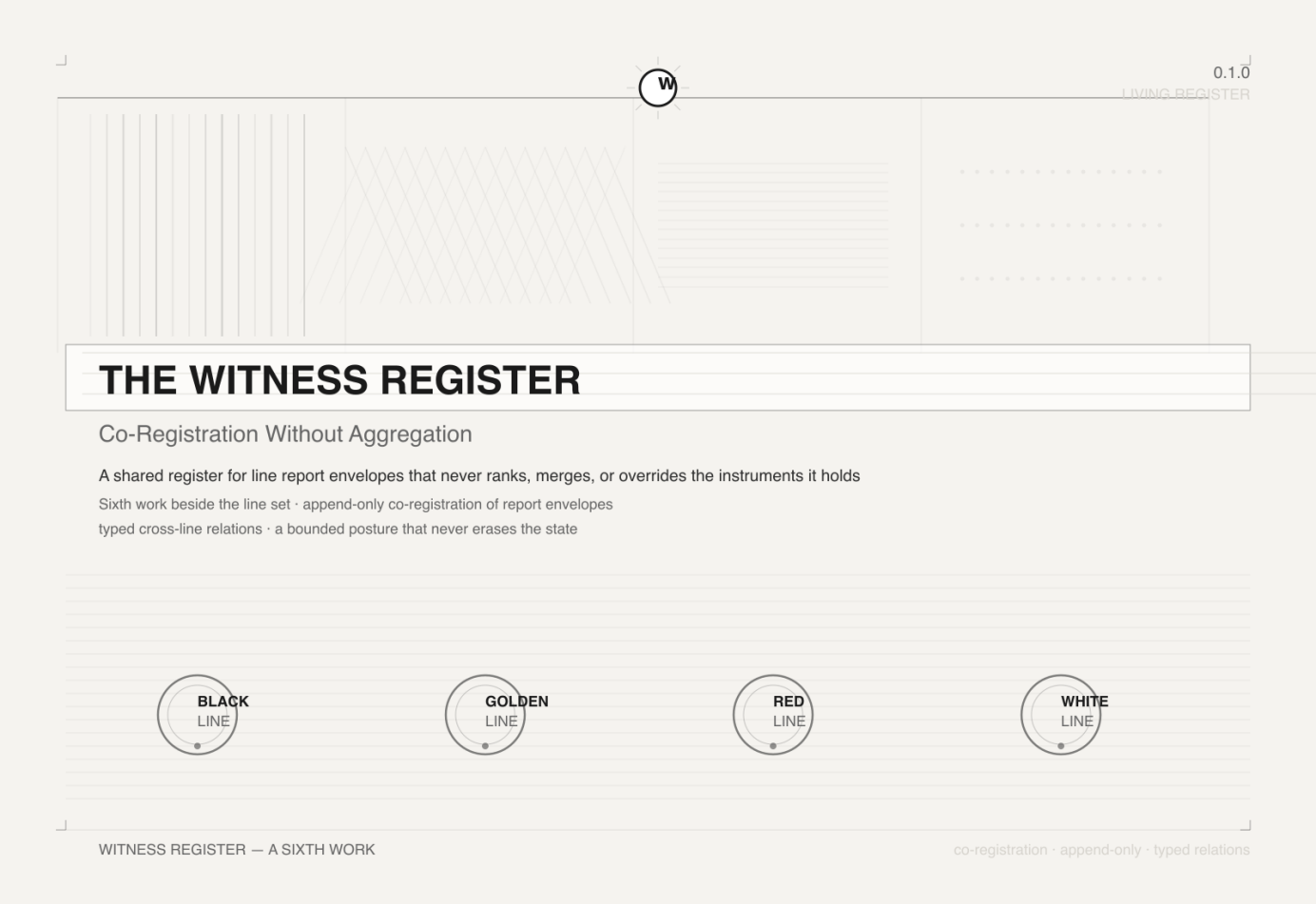


Figure 1: The Witness Register cover plate: four vertical ledger columns with distinct greyscale hatch textures run in parallel without converging; a horizontal register band carries the title across all four; four seal circles mark the instruments below, and the witness mark sits above and alone. The cover uses only the register's greyscale palette — no colour that would affiliate it with any one instrument.

Four independent instruments — the line works — each answer their own substantive question about a subject in their own vocabulary, and each ends in a status. A reader holding all four reports faces a temptation with a long institutional history: collapse them into one number. Average the verdicts, rank the instruments, let the loudest status become the state.

The 2026-07-29 design review of the collected line set ("The Space Between the Lines", an external reviewer, with an analytic reader — unpublished correspondence, answered in this repository's `docs/correspondence.md`) names what that collapse destroys. A selected status is a SAFE PROJECTION of a richer state, and the projection must not become the whole state: strong support and strong resistance co-present are not the same as no evidence, though both can project to the same cautious middle. A protected absence is not missing evidence to be mined. A block that no other strength may buy back disappears the moment scores are allowed to compensate one another.

The review's proposal is a missing layer, not a fifth verdict: a SHARED WITNESS REGISTER that co-registers each line's report envelope, stores cross-line relations as separate records, keeps history append-only, and must never rank, average, merge, score, or override the lines. Its mottos: *precedence without information destruction*; *no crown without return.*

This work is that register, built as a sixth work standing beside the set. It is deliberately not a line. It has no colour, no substantive question of its own, and no verdict. Each line already exports one common report envelope under the published schema string `line.report-envelope/1.0` — a digest pointer to its complete native report, its identity, subject, review date, registry provenance, native status in its own vocabulary, and its non-claims. The register accepts those envelopes by value — the schema string is aligned across repositories by published convention, never by import — and holds them beside one another without reading past their covers.

"'{=latex}

“‘

The paper proceeds as follows. Section sec. 3 describes the register's architecture — the envelope contract, relations, chain, and projection. Section sec. 6 documents how the register is built and operated. Section sec. 4 states the instrument formally, from the definition of an envelope record (Definition 1) through the projection invariants (Proposition 2). Section sec. 5 situates the work in its intellectual lineage. Section sec. 8 presents two worked examples — the co-registration of four instruments and the same-subject return contract — together with the 3×3 battery. Section sec. 9 states the epistemic boundaries and non-claims, and Section sec. 10 closes.

3 The Design

3.1 Non-sovereignty, stated first

The register's defining properties are refusals, enforced in code and restated as non-claims in every module: it never imports a line package; it never parses, compares, ranks, averages, or merges any line's `native_status`; it never auto-creates a category; it never infers consent or permission; it never mutates or rewrites history; and it emits no score other than one bounded posture, and that only on request. Each line remains the sole authority over its own vocabulary. The register witnesses; it does not judge.

3.2 The envelope contract

Intake is a shape check over the published `line.report-envelope/1.0` payload: schema string exact, line identity non-blank, review date a real ISO calendar date, registry digest and report pointer 64 lowercase hex, snapshot references non-blank strings, and a non-empty set of non-claims — an envelope without its instrument's boundary can quietly outgrow what the instrument was allowed to say. `native_status` may be any JSON-compatible value and is stored verbatim. Nothing is rejected silently: refusal returns typed issues naming every defect. Accepting an envelope asserts nothing about the truth of its report.

3.3 Relations describe; they never replace

Cross-line structure lives in separate relation records over the envelopes' report pointers: non-compensatory block, unresolved dependency, protected absence, directional tension, unclassified observation, return due, cannot-compare, and agreement. A relation keeps its surfaces apart — support references and resistance references are distinct fields — so a conflict survives as two sides rather than a summary. An empty `human_decision_ref` means NOT_RECORDED, a first-class honest value, never a default verdict.

Inputs that fit no category are held raw, outside every alphabet, with a stated reason. Promotion into the relation vocabulary requires a non-empty human decision reference and yields a new relation that links back to the original holding, which stays in history verbatim.

3.4 Append-only, sealed, and returned to

States are sealed by a digest over their complete canonical content — including record order — and chained by `prior_ref`. An update must carry every prior record unchanged; the prior seal is re-derived from live content first, so in-place tampering fails closed (formalized in the formal core; construction documented in the method section). Return contracts record what must come back, from whom, under what trigger, and with what observable acceptance condition; a completed return is a new record beside the open one and closes only the verified part, the remainder keeping its trigger. The review's 3×3 canonical witness cases ship as an executable battery whose checks are themselves proven able to reject (described in the examples section).

3.5 The posture that cannot travel alone

On request, for one declared next use, the register projects `-1 | 0 | +1`. The invariants are non-compensatory and enforced in code, driven only by relation records: an unresolved block forces `-1`; a protected absence forbids `+1` unconditionally; an unmet return forbids `+1` until the return condition is met or a referenced human decision rescopes; an empty register is `-1`, because nothing to witness is not permission. The symbols are interface values, not the ontology: every projection carries the digest of the state that earned it and the reasons it holds, and a held `0` is a structured enumeration, never a vague middle (formalized in the formal core). Three first-pass measures — relation fidelity, return recoverability, premature crowning rate — evaluate the register's own bookkeeping, never the truth of any report.

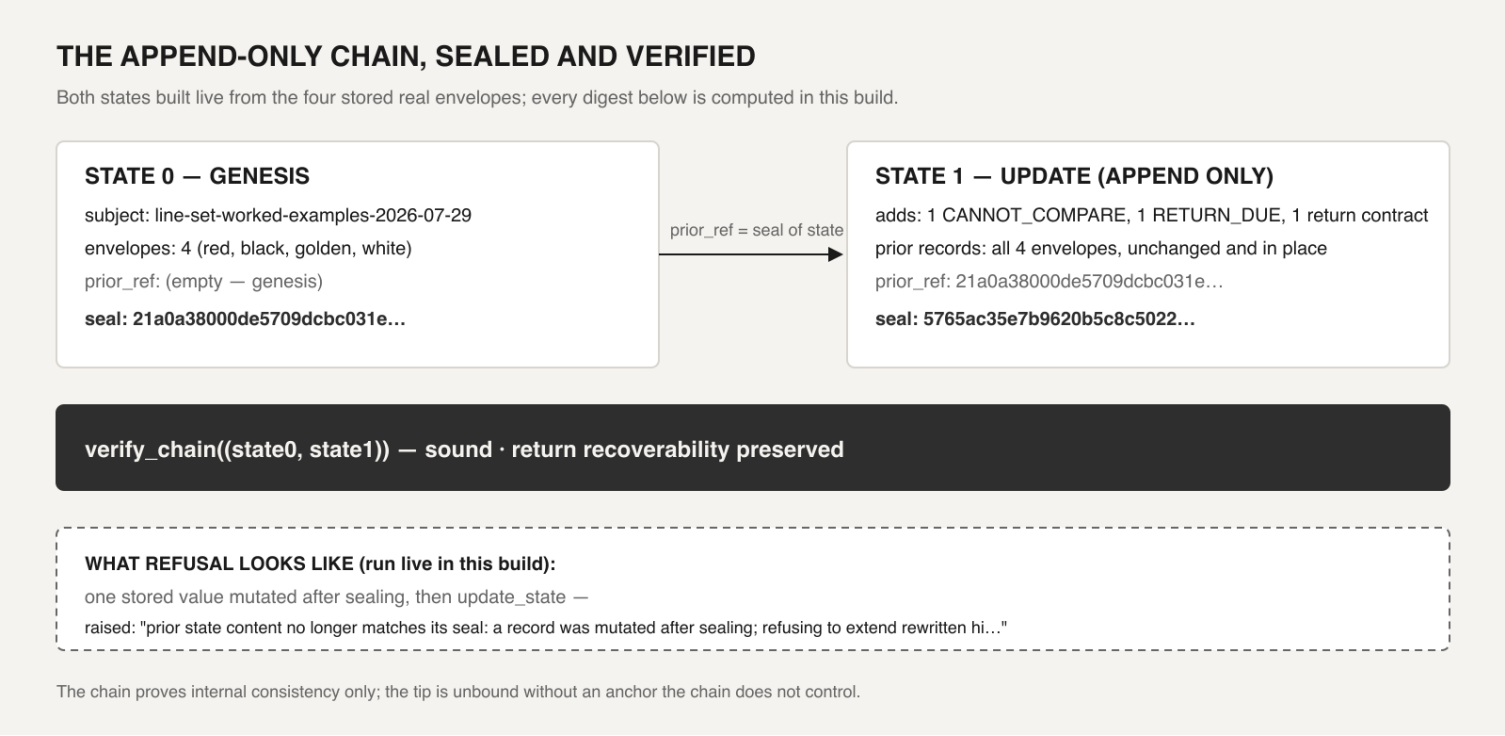


Figure 2: The worked two-state chain, built live from the four stored real envelopes at figure-build time: the genesis seal, the update's prior_ref pointing at it, verify_chain's verdict on the pair, and — run in the same build — the exact refusal update_state raises when one stored value is mutated after sealing. Chain integrity is internal consistency only; the tip is unbound without an anchor the chain does not control.

4 The Formal Core

The register's behaviour is small enough to state completely. Every definition and proposition below is written from the module it describes and bound to the running package by a named test in `tests/test_formalism.py`; the numbers the reader sees are assigned by the render toolchain from document order, and none is written in this source.

Definition 1 (Envelope record). An accepted envelope is the frozen record e = (schema_version, line_id, subject_id, review_date, r with exactly those ten fields, in order — the published `line.report-envelope/1.0` shape, held verbatim. native_status is opaque here: the register stores it and never parses, compares, ranks, averages, or merges it. Intake accepts strict JSON only; a payload carrying `NaN` or `Infinity` in its status is refused with a typed issue, because a digest over text that is not JSON would seal a non-interchangeable value.

Definition 2 (Witness state and chain). A witness state is the frozen record X = (subject_id, review_moment, envelopes, relations, with exactly those eight fields, sealed by state_digest = SHA-256 over the canonical JSON of every other field including record order. States form a chain: a genesis state carries an empty prior_ref; every every other state carries the digest of the state it extends and must contain every prior envelope record (see Definition 1) unchanged and in place before any addition.

Proposition 1 (Append-only, fail-closed). The update function re-derives the prior witness state's (see Definition 2) seal from its live content before extending it, so a record mutated after sealing — even a value buried inside an opaque `native_status` — raises rather than being carried forward; `verify_chain` applies the same checks to a stored chain after the fact and never repairs anything; and `seal_tip` refuses to hand out an anchor digest for content that no longer matches its seal. Projection applies the identical re-derivation, so no posture is ever stamped onto rewritten history.

Definition 3 (Projection and zone). For a witness state X (see Definition 2) and a declared next use u (blank u raises: no posture without a declared use), the projection is $P(X, u) \in \{-1, 0, +1\}$, always carrying the digest of the exact state that earned it and at least one reason. The zone is computed from relation records only, in fixed precedence: any non-compensatory block with no recorded human decision forces -1; an empty state is -1, because nothing to witness is not permission; any hold — a protected absence, an outstanding return, an unresolved tension, dependency, incomparability, or unreviewed observation, or an unreviewed unclassified holding — caps the value at 0; and $+1$ is reachable only when at least one envelope record (see Definition 1) exists and nothing recorded forbids the use. Non-compensatory means the same thing here that Proposition 2 states: a single block forces -1 under any volume of agreement, and no decision reference lifts it.

Proposition 2 (Non-compensatory means non-compensatory). No volume of recorded agreement buys back a blocked route in the projection (see Definition 3), a protected boundary, or an unmet return: a single unresolved block forces −1 under any number of agreement relations; a protected absence caps the posture at 0 past every decision reference, including the one offered at projection time; and a return contract with no verified return — or a verified return with a named remainder — holds the posture until the return condition is met or a human decision explicitly rescopes it, and a whitespace-only verification is refused at construction rather than counted as met.

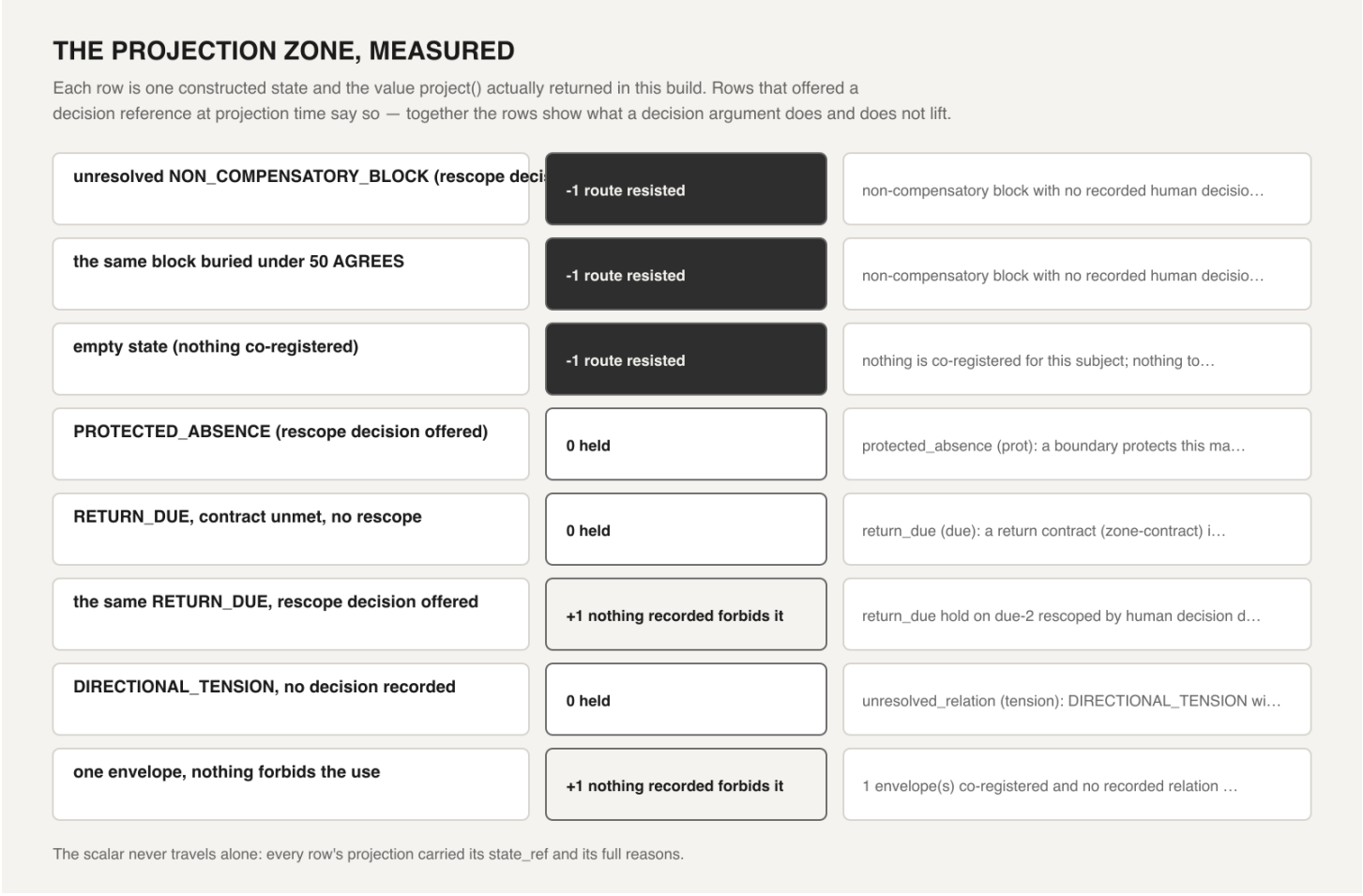


Figure 3: The projection zone measured: each row is one constructed state and the value project() actually returned in this build, with a human decision reference offered at projection time so the rows also show what a decision argument does not lift — it never resolves a block and never lifts a protected absence. A non-compensatory block forces -1 alone and under fifty AGREES relations alike; an empty register is -1, not permission; every hold row is 0 with its reason; +1 appears only where an envelope exists and nothing recorded forbids the use.

Proposition 3 (No auto-categories, no manufactured decisions). An observation that fits no current category is held raw, outside every alphabet, and holding is not permission: an unreviewed holding caps the projection (see Definition 3) at 0. Promotion into the relation vocabulary requires a non-empty human decision reference — blank and whitespace-only references are refused — and the promoted relation links back to the held record, which is never rewritten.

4.1 The worked co-registration

The repository carries four real envelopes under `data/envelopes/`, one per line, each generated by running that line's own public API in its own repository on 2026-07-29 and stored by value with a provenance record. They describe four *different* worked subjects — each line's own shipped example — so the honest structure over them is an incomparability relation and an open return contract naming what a same-subject co-registration would require. `tests/test_worked_example.py` intakes all four unmodified, builds the two-state chain, and measures: chain verification clean (Proposition 1), return recoverability 1.0, relation fidelity 1.0, and the posture for treating the co-registration as a live subject record held at 0 with both records named in the reasons. The held posture is the demonstration: real data, honest relations, and a register that declines to crown them. The holding is not permission — it is the same non-auto-category rule Proposition 3 states: an observation that fits no current category is held raw, and holding caps the projection at 0.

The return was then met the way the contract said it must be: four further envelopes, each line's real evaluator run over ONE declared work — witness_register 0.1.0 itself, with registrar-authored inputs whose provenance is recorded beside the records. The instruments answered in their own vocabularies: a declaration-coverage `ALIGNED`, an honest `outside_scope` for the one action asked about, two directional `TOWARD` readings with seven `NOT_OBSERVED`, and an absence ledger naming the unobserved tip-anchor dependency and one `UNRESOLVED` question. Completing the contract lifts exactly the `return_due` hold and nothing else — the incomparability relation still holds the first chain at 0 — and the same-subject state's own posture is also held at 0, because the honest reading of the ledger's open question enters as an unresolved dependency. Three favorable readings and one open question is a held posture, not a crown; `tests/test_same_subject.py` measures every sentence of this paragraph.

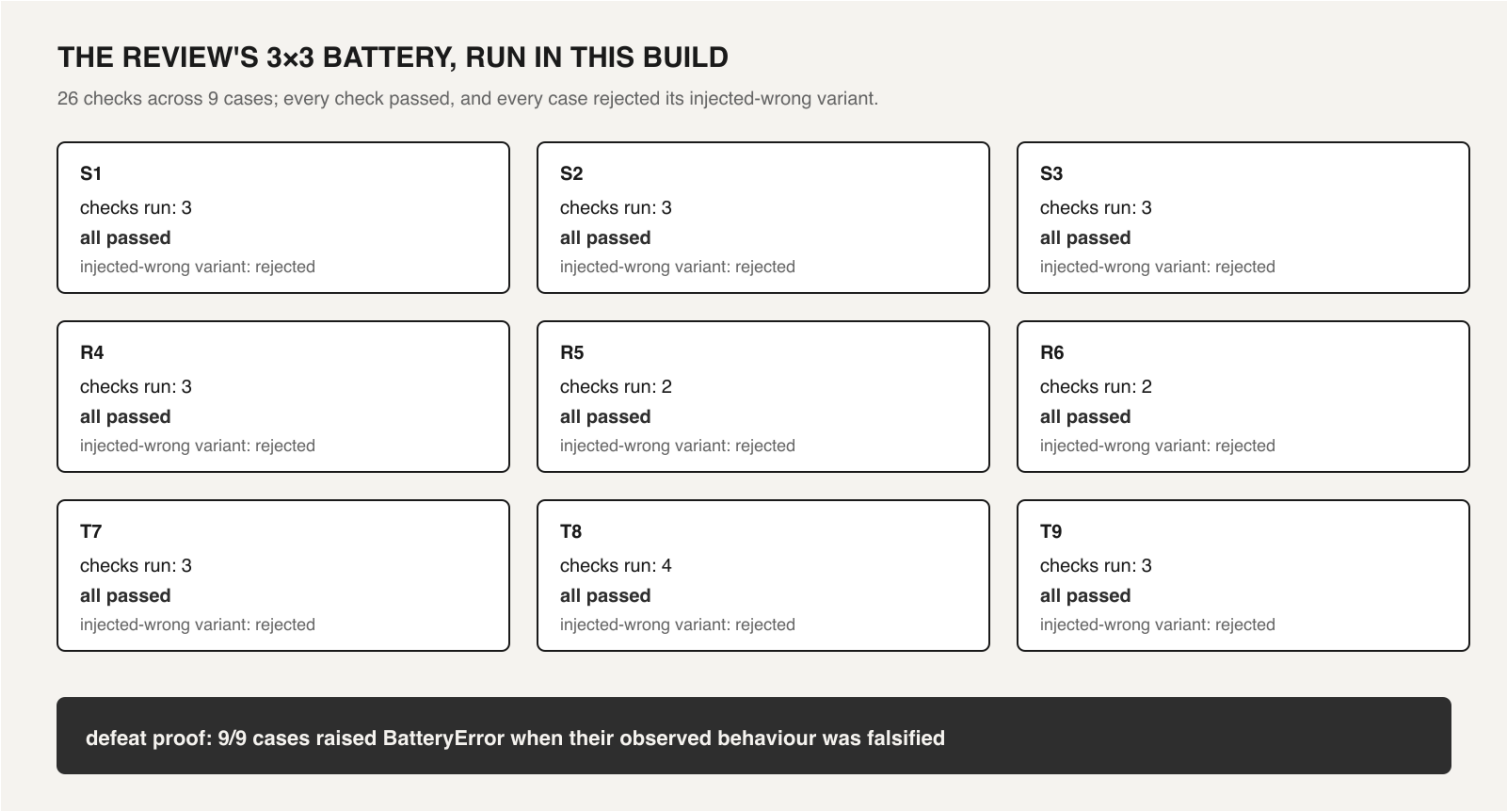


Figure 4: The design review's 3×3 canonical witness cases as the shipped battery, run at figure-build time: every check passed on the real register, and — measured in the same build — every case raised BatteryError when its observed behaviour was deliberately falsified, so a green grid is evidence the checks can fail, not only that they passed.

5 Scholarship and Intellectual Lineage

The register's mechanisms are small, and none of them is new. Each design element below names the tradition it borrows from, what exactly is taken, and where the register deliberately stops short of the cited work's ambition. The scholarship does not validate the register's postures — the register makes no judgments to validate. The cited traditions are cited for how they keep books, not for any claim about the truth of what the books record. Every bibliographic record was verified against Crossref, the RFC Editor, or the W3C on 2026-07-29 before being cited; nothing here is cited from memory.

5.1 Linked timestamping and the sealed chain

The witness state chain — each state carrying a digest over its complete canonical content and the digest of the state it extends — is the linking scheme of Haber and Stornetta [Haber and Stornetta, 1991], who showed that chaining document digests makes the *order and content* of a record series tamper-evident without trusting the record keeper's clock. Their scheme addressed the problem of certifying when a document was created or last modified: the linking step — each certificate including a hash of the previous — makes retroactive insertion or reordering detectable, because any change to an earlier link breaks every subsequent digest. The register takes exactly that: internal tamper evidence inside the chain.

What the register does not take is the further apparatus. Haber and Stornetta's scheme includes distributed trust mechanisms — linking with other clients, publishing hash values in widely-witnessed media — to guard against a dishonest time-stamping service issuing backdated certificates that form an internally consistent but temporally false chain. The register's tip is unbound for precisely this reason: an append-only chain guarantees that history inside it cannot be rewritten undetected, but nothing inside the chain can detect a discarded tip. A holder of the whole chain may present an earlier state as current, and the chain itself cannot know. The limitation is stated rather than solved — see the limits section (sec. 9) — and `seal_tip` hands out the value such distributed infrastructure would anchor.

The digest-over-canonical-content discipline itself belongs to the hash-authentication tradition Merkle formalized for signatures and trees [Merkle, 1990]. The register's `state_digest` is a single SHA-256 hash over canonical JSON — a degenerate one-element tree — and the re-derivation check before every chain extension is the Merkle-verification idiom applied to the smallest possible unit: a state sealed against itself. The tree is absent because the register holds a linear chain rather than a branching structure of many leaves; the verification discipline is the same.

5.2 Certificate Transparency and the tip-unbound problem

The tip-unbound limitation is not a defect of the register's design; it is the log-consistency problem that Certificate Transparency (CT) made explicit at infrastructure scale. CT's original specification [Laurie et al., 2013] built public append-only logs for X.509 certificates, and the current version [Laurie et al., 2021] refines the same architecture. A CT log is a Merkle tree hash chain that grows monotonically; monitors verify that the log is append-only, and auditors check that particular certificates appear. The critical insight — the one the register restates at a much smaller scale — is that a log can be internally consistent while presenting different views to different observers. A log operator who serves a truncated view to one monitor and the full chain to another has produced two consistent proofs from different states, and neither proof by itself reveals the divergence.

CT's answer is infrastructure the register does not have and does not claim: gossip protocols between monitors, multiple independent logs, and external auditing. These mechanisms move the consistency check *between* observers, outside the log. The register's `seal_tip` is the value such an infrastructure would anchor; until an anchor exists in a system the chain does not control, chain integrity is a claim about internal consistency only. The register states this limitation rather than implying it has been solved, and the scholarship makes the provenance of the limitation explicit: it is not a missing feature but a known structural boundary of append-only logs, acknowledged at full scale by the CT working group.

5.3 Non-compensatory decision rules

The projection's invariants — a block that no volume of agreement buys back, a protected absence no strength outweighs, precedence in a fixed order — are non-compensatory decision rules in the sense surveyed by Fishburn [Fishburn, 1974]. Fishburn's survey of lexicographic orders formalizes structures in which one criterion's verdict cannot be traded against quantities of another: the first-ranked attribute decides, and lower-ranked attributes are consulted only when higher-ranked ones fail to discriminate. The register's projection precedence — unresolved block before empty state before any hold, with a single block forcing `-1` under any volume of `AGREES` relations — is a small instantiation of that structure, deliberately chosen because compensatory aggregation is exactly the averaging of instruments the design review forbids.

What the register does not borrow is the utility apparatus. Fishburn's survey is grounded in decision theory and utility maximization: lexicographic orders are studied as preference structures, and the question is whether a lexicographic preference can be represented by a real-valued utility function (it cannot, in the general case — a fact the register exploits structurally). The register's posture is not a utility, not a preference, and not optimal by any criterion. It is a bounded interface value carrying the digest

of the state that earned it, the reasons it holds, and nothing more. The borrowing is confined to the *shape* of non-compensatory ordering; the theoretical apparatus that accompanies that shape in decision theory is set aside.

The same point applies in the other direction. Fishburn's survey covers lexicographic orders over multidimensional attribute spaces; the register's precedence is a flat list of seven checks, not a hierarchy of attributes. The register does not implement a lexicographic decision rule in the formal sense — it implements a precedence list whose shape resembles one, and the resemblance is the reason for the citation. The formal claim is stated in the formalism section (sec. 4): non-compensatory means non-compensatory, and Proposition 2 is enforced in code rather than asserted in prose.

5.4 Provenance as description, not rewriting

The rule that relation records describe stored envelopes without rewriting them — disagreement kept as two surfaces, authority fields honest about `NOT_RECORDED`, a holding stored raw beside its later classification rather than replaced by it — follows the posture of the W3C provenance data model [Moreau and Missier, 2013]. PROV-DM types what was derived from what, and by whom, as records *about* artifacts rather than edits *to* them. An entity record does not overwrite the entity; a derivation does not delete the thing derived from. The register's relation records are built to the same discipline: an `AGREES` relation does not merge the agreed-upon envelopes, a `BLOCK` does not delete what it blocks, and a `RETURN_DUE` does not presume what the return will contain.

The register's vocabulary is far smaller than PROV's and deliberately so. PROV includes entities, activities, agents, derivations, generations, usage, attribution, association, delegation, communication, and a formal constraint language over them. The register has eight relation types entered by people, no inference engine, and no automatic derivation. The narrowing is not a criticism of PROV — it is a scope decision: the register stores what people record, and the people who record it are responsible for the categories they use. A provenance vocabulary large enough to model scientific workflows, software builds, and data pipelines would be overbuilt for eight relation types over four report envelopes.

The structural discipline — records about artifacts that never mutate the artifacts — is the single point the register takes from PROV, and the citation acknowledges the lineage without claiming equivalence.

5.5 The envelope as a boundary object

Star and Griesemer's standardized forms [Star and Griesemer, 1989] are artifacts robust enough to travel between communities while staying locally interpretable. The paper's canonical examples are standardized data-entry forms that amateur collectors and professional zoologists filled and read without adopting one another's theories: the form carries information across a boundary without requiring either side to convert to the other's conceptual scheme. The `line.report-envelope/1.0` record is built to that specification: ten fields any line can export and the register can hold, with `native_status` staying in the exporting instrument's own vocabulary precisely so that transport never becomes translation.

Star and Griesemer's typology distinguishes four kinds of boundary object: repositories, ideal types, coincident boundaries, and standardized forms. The envelope is a standardized form — a fixed shape, filled in the same way each time, whose function is to carry information across a boundary without negotiating it away. The register's refusal to parse, compare, rank, average, or merge `native_status` is the operationalization of that design choice: the field stays in the line's own vocabulary because the register is a transport surface, not a translation layer.

Star later objected to how the concept had been taken up, in particular to its application to any object that happens to sit between two groups, detached from the infrastructural and standardizing work the original study was about [Star and Griesemer, 1989]. Her caution applies here and is adopted: not everything that sits between groups is a boundary object, and the envelope earns the name only while each line remains authoritative over its own field. Four Python packages and one shared register are not the Museum of Vertebrate Zoology, and the register's relationship to the lines is not a negotiation between amateur collectors and professional zoologists. What is taken from the 1989 paper is a design move — a standardized form whose local fields stay local — and the concept is not doing evidentiary work beyond that.

5.6 What the borrowings borrow and what they leave

The lineage-to-wire map is deliberately asymmetric:

Scholarly tradition	Register's operational slice	Boundary preserved
Linked timestamping [Haber and Stornetta, 1991]	chained digests for internal tamper evidence	distributed trust mechanisms; the tip is unbound
Hash authentication [Merkle, 1990]	digest-over-canonical-content seals; re-derivation check before extension	the tree structure; the register holds a linear chain

Scholarly tradition	Register's operational slice	Boundary preserved
Certificate Transparency [Laurie et al., 2013, 2021]	the log-consistency problem stated as the tip-unbound limitation	gossip, multiple logs, external auditing — infrastructure the register does not have
Non-compensatory decision rules [Fishburn, 1974]	lexicographic precedence shape: a single block forces `-1` under any volume of agreement	the utility apparatus; the posture is not a utility, not optimal, and not a preference
Provenance data model [Moreau and Missier, 2013]	records *about* artifacts that never mutate the artifacts	the full PROV vocabulary, inference engine, and constraint language
Boundary objects [Star and Griesemer, 1989]	a standardized form whose local field stays local; transport without translation	the institutional ecology; the register is not the Museum of Vertebrate Zoology

The table is a design map, not a claim that six citations capture these traditions. Its purpose is to make the borrowing inspectable and the non-borrowing equally explicit: each row states what was taken and where the taking stops.

5.7 What the lineage does and does not license

Citing these works situates the register; it does not borrow their authority. None of these authors claims that following a bookkeeping discipline guarantees a correct result, and neither does this register. The lineage explains *why* each mechanism is worth making inspectable; the formal core and further propositions state *what the evaluator can actually check*, which is only whether the declared invariants hold — never whether the underlying reports are true, the relations are warranted, or the posture is wise.

The register's scholarship is therefore scoped to mechanisms: linked timestamping for the chain, transparency logs for the tip-unbound limitation, non-compensatory decision rules for the projection invariants, provenance records for relation discipline, boundary objects for the envelope contract. Each borrowing is named, its boundary is stated, and the register's refusal to reach beyond those boundaries is the point. A register that keeps books honestly is not a register that judges wisely, and the scholarship section's task is to make that separation legible rather than to collapse it.

6 Method: How the register is built and operated

The register is small — a few hundred lines of Python — and the design section states what it does. This section states how it is built, how it is operated, and how it is verified. The formalism (sec. 4) restates the same machinery as objects and rules; this section describes the construction.

6.1 Building the chain

The register lives as a Python package — `witness_register` — with no import dependencies on any line package. It never imports `black_line`, `golden_line`, `red_line`, or `white_line`. Each line already exports one common report envelope under the published schema string `line.report-envelope/1.0` (Definition 1), and the register accepts those envelopes as JSON files stored by value under `data/envelopes/`.

Intake is a shape check over the published payload: the schema string must be `line.report-envelope/1.0` exactly, the line identity must be non-blank, the review date must be a real ISO calendar date, the registry digest and report pointer must be 64 lowercase hexadecimal characters, snapshot references must be non-blank strings, and the non-claims set must be non-empty. `native_status` may be any JSON-compatible value and is stored verbatim — the register never parses, compares, ranks, averages, or merges it.

Nothing is rejected silently: refusal returns typed issues naming every defect. A payload carrying `NaN` or `Infinity` is refused because a digest over text that is not JSON would seal a non-interchangeable value.

Two states form a chain: a genesis state carries an empty `prior_ref`; every other state carries the digest of the state it extends. The update function re-derives the prior state's seal from its live content before extending it, so a record mutated after sealing — even a value buried inside an opaque `native_status` — raises rather than being carried forward. The chain is append-only and fail-closed (Proposition 1).

6.2 Recording relations

Cross-line structure lives in separate relation records over the envelopes' report pointers. Each relation names its type — non-compensatory block, unresolved dependency, protected absence, directional tension, unclassified observation, return due, cannot-compare, or agreement — and carries distinct support and resistance reference fields, so a conflict survives as two sides rather than a summary.

An empty `human_decision_ref` is the honest value `NOT_RECORDED`, never a default verdict. Inputs that fit no current category are held raw, outside every alphabet, with a stated reason. Promotion into the relation vocabulary requires a non-empty human decision reference and yields a new relation that links back to the original holding, which stays in history verbatim (Proposition 3).

Return contracts are a special case: they record what must come back, from whom, under what trigger, and with what observable acceptance condition. A completed return is a new record beside the open one and closes only the verified part, the remainder keeping its trigger.

6.3 Computing the projection

On request, for one declared next use, the register projects `-1 | 0 | +1`. A blank use raises — no posture is issued without a declared use. The zone is computed from relation records only, in fixed precedence (Definition 3):

1. Any non-compensatory block with no recorded human decision forces `-1`.
2. An empty state is `-1`, because nothing to witness is not permission.
3. Any hold — a protected absence, an outstanding return, an unresolved tension, dependency, incomparability, or unreviewed observation, or an unreviewed unclassified holding — caps the value at `0`.
4. `+1` is reachable only when at least one envelope exists and nothing recorded forbids the use.

Every projection carries the digest of the state that earned it and at least one reason. The symbols are interface values, not the ontology: a held `0` is a structured enumeration, never a vague middle.

6.4 The executable battery

The review's 3×3 canonical witness cases ship as an executable battery in `tests/test_battery.py`. Each case is a constructed state and a predicate the projection must satisfy: three positive cases (the register should pass), three negative cases (it should hold), and three adversarial cases (it should reject). Every case is run against the real projection function at test time; the battery also includes a falsification pass that plants deliberate errors and confirms the check would have failed, so a green grid is evidence the checks can fail, not only that they passed (fig. 4).

Three first-pass self-measures — relation fidelity, return recoverability, and premature crowning rate — evaluate the register's own bookkeeping, never the truth of any report. They are computed from the live chain and are reported alongside the battery results; they carry no assertion about safety, correctness, or completeness.

7 Further propositions: chain, return, and deterministic projection

The three propositions below extend the formal core of sec. 4. Each states a property of the running code; each is verified by the test named in the binding table that follows.

Proposition 4 (Chain verification completeness). When a witness chain is constructed from observed events by successive calls to `update_state` (see Definition 2), every event in the chain is verifiable against the register's evidential record: no event can enter a chain without a corresponding register entry, and `verify_chain` checks every state's digest (Proposition 1), linkage (`prior_ref`), and record preservation for envelopes, relations, unclassified holdings, and return contracts. A state that dropped, reordered, or altered any prior record fails verification with a message naming the violation; verification never repairs.

Proposition 5 (Return contract enforcement across chains). When the same subject appears in multiple chains, the return-contract projection constrains the relationship between those chains. For a subject S appearing in chains A and B, the projection (Definition 3) applies the same rules: any `RETURN_DUE` relation with no recorded human decision forbids +1 until the return condition is met. A contract recorded in one chain does not automatically bind another, but within a single state the projection respects every open return recorded there, and meeting a contract requires a verified return record (Proposition 2) — a partial return with a non-empty remainder does not close the obligation. Return contracts are records of obligation, not promises of approval: the honest outcome of a return may be that the material stays held.

Proposition 6 (Projection determinism). Given the identical witness register state X (see Definition 2) and the identical declared next use u, `project(X, u)` produces the identical `Projection` value — same `value`, `state_ref`, and `reasons` — every time. The projection function is a pure computation over the state's sealed content: it re-derives the state's digest from live content before projecting (Proposition 1), checks for unresolved blocks, empty state, and holds in a fixed precedence (Definition 3), and returns a deterministic result. The digests are deterministic in their own right; the projection carries the digest of the exact state that earned it. Determinism is a property of the code path; it does not make the underlying records true or the posture warranted.

7.1 Formalism-to-test bindings

Every proposition above is verified by named tests in the package's suite; the table below binds each result to the test that would fail if the code stopped satisfying it. Each row is keyed on the block's *label*, not on its number, and the label renders as the number the reader sees.

Two binding tests police the table. `tests/test_formalism_bindings.py::test_binding_tables_bind_every_declared_block` fails if the set of row labels stops matching the set of labels declared in this section, and `tests/test_formalism_bindings.py::test_every_binding_row_names_an_existing_test` fails per row if any row's verifying-test cell names no test or names one that does not exist. Neither checks that a named test is a *good* test, only that every declared block is bound to one that exists. The boundary column restates what each result does *not* claim.

Proposition	Statement essence	Verifying test	Boundary
Proposition 4	every event in a chain is verifiable against the register's evidential record; verification never repairs	`tests/test_formalism_bindings.py::test_chain_verification_checks_every_state`	chain internal consistency, never truth of events
Proposition 5	same-subject return contracts constrain the projection; partial returns do not close the obligation	`tests/test_formalism_bindings.py::test_return_contract_constrains_projection_across_chains`	records of obligation, never promises of approval
Proposition 6	identical state and use produce identical projection; deterministic code path	`tests/test_formalism_bindings.py::test_projection_is_deterministic_for_identical_inputs`	determinism of the code path, not truth of records

The bindings are themselves code behaviour: they show which claims the suite would catch, not that the register's posture is wise or well grounded.

8 Worked Examples

Two worked examples exercise the register over real data: the co-registration of four instruments over different subjects, and the same-subject return contract. Both use envelopes generated by running each line's own public API on 2026-07-29 and stored by value with provenance records in `data/envelopes/`. The examples are executed at test time — `tests/test_worked_example.py` and `tests/test_same_subject.py` — not described from memory.

8.1 Different subjects: the co-registration

The four envelopes describe four *different* worked subjects — each line's own shipped example — so the honest structure over them is an incomparability relation and an open return contract naming what a same-subject co-registration would require.

`tests/test_worked_example.py` intakes all four unmodified, builds the two-state chain, and measures: chain verification clean (Proposition 1), return recoverability 1.0, relation fidelity 1.0, and the posture for treating the co-registration as a live subject record held at 0 with both records named in the reasons.

The held posture is the demonstration: real data, honest relations, and a register that declines to crown them. The holding is not permission — it is the same non-auto-category rule Proposition 3 states: an observation that fits no current category is held raw, and holding caps the projection at 0.

The chain figure (fig. 2) shows the live two-state construction; the battery figure (fig. 4) shows the 3×3 falsifiable checks.

8.2 Same subject: the return contract

The return was then met the way the contract said it must be: four further envelopes, each line's real evaluator run over ONE declared work — witness_register 0.1.0 itself, with registrar-authored inputs whose provenance is recorded beside the records.

The instruments answered in their own vocabularies: a declaration-coverage `ALIGNED`, an honest `outside_scope` for the one action asked about, two directional `TOWARD` readings with seven `NOT_OBSERVED`, and an absence ledger naming the unobserved tip-anchor dependency and one `UNRESOLVED` question.

Completing the contract lifts exactly the `return_due` hold and nothing else — the incomparability relation still holds the first chain at 0 — and the same-subject state's own posture is also held at 0, because the honest reading of the ledger's open question enters as an unresolved dependency. Three favorable readings and one open question is a held posture, not a crown; `tests/test_same_subject.py` measures every sentence of this paragraph.

The return contract demonstrates the register's core discipline: a non-compensatory invariant (Proposition 2) means that one open question prevents the posture from reaching +1, regardless of how many favorable readings sit beside it. This is not a defect — it is the design.

8.3 The 3×3 battery as evidence

The review's 3×3 canonical witness cases ship in `tests/test_battery.py` as a battery whose checks are themselves proven able to reject. Each cell is a constructed state and a predicate the projection must satisfy; the falsification pass plants a deliberate error in every predicate and confirms the check would have failed. A green grid that could not reject is not evidence; the battery's falsification pass is what makes the green grid informative (fig. 4).

The three positive cases exercise the projection over states where the register should reach +1: a single envelope with no recorded relation forbidding the declared use, a pair of envelopes with a recorded agreement, and a state where a return contract has been met. The three negative cases exercise states where the register should hold at 0: an unreviewed holding, a protected absence, and an outstanding return. The three adversarial cases exercise states where the register should force −1: a non-compensatory block, an empty register, and a tampered seal.

Those nine cases are the same battery shown in fig. 4; the figure's grid is rendered live from `run_battery()` at build time, so the plate and these sentences cannot drift apart.

8.4 What the examples do not establish

The co-registration uses four envelopes from one date. It does not demonstrate the register's behaviour over time, over a changing roster of instruments, or over adversarial input designed to exploit the projection rules. The same-subject return contract uses the registrar as its own subject — a deliberate self-application — and does not demonstrate the contract with an external subject or a subject who declines to return. The battery exercises nine constructed states; it does not enumerate the full state space. These limits are developed in sec. 9.

9 Limits and Epistemic Boundaries

Stated plainly, in the set's tradition of writing the edge of the claim.

The tip is unbound. An append-only chain guarantees that history inside it cannot be rewritten undetected — but nothing inside the chain can detect a discarded tip. A holder of the whole chain may present an earlier state as current. The register hands out the tip digest for external anchoring and solves nothing beyond that; until an anchor exists in a system the chain does not control, chain integrity is a claim about internal consistency only (Proposition 1). Chain verification (Proposition 4) confirms every state links to its predecessor and every digest matches, but it verifies the presented segment — it cannot detect whether a newer state exists outside it.

Intake is a shape check. A structurally perfect envelope can point at a fabricated report. The register's acceptance asserts published shape, never truth; verifying a report means returning to the exporting line, which is the point of storing pointers instead of copies (Definition 1).

Relations are authored. The register enforces invariants over relation records, but the records themselves are entered by people (or by lines' own tooling upstream). A missing relation is invisible to projection: if no one registers the block, the posture will not show it. This is the honest cost of refusing to parse native vocabularies — the register cannot infer what it was never told, and it declines to guess (Definition 3).

The posture is not a decision. `+1` means nothing recorded forbids the declared next use; it is not endorsement, safety, quality, or permission from any boundary owner. `-1` resists a route; it does not condemn an aim. `0` is an enumeration of holds, not a compromise. Any use of these symbols without their state reference and reasons has already violated the design (Proposition 2). The posture is deterministic for identical state and use (Proposition 6), but determinism is a property of the code path — it does not make the underlying records true or the posture warranted.

The measures measure the register. Relation fidelity, return recoverability, and premature crowning rate evaluate bookkeeping. A register can score perfectly on all three while every underlying report is wrong; that would be the lines' failure to catch, each in its own vocabulary, and the register's success at not hiding it.

The scholarship is scoped to mechanisms. The cited traditions — linked timestamping, transparency logs, non-compensatory decision rules, provenance records, boundary objects — are cited for how they keep books, with every bibliographic record verified before use. None of them underwrites any judgment, because the register makes none, and the borrowings stop where each tradition's larger ambitions begin (distributed trust, gossip infrastructure, utility theory, inference engines).

9.1 Adversarial declarations

Because every input is self-declared — envelopes carry pointers to reports, relations are authored, and declared next uses choose their own scope — the instrument can be gamed by construction, and the honest response is to demonstrate the attacks rather than deny them. Each of the following was exercised against the real register.

Malicious envelope injection. An adversary can submit a structurally valid envelope whose `report_ref` points to a fabricated or doctored report the register has no access to verify. The register stores the pointer and seals it into chain state; it never opens the pointed-to report. A stored envelope with a valid shape certifies nothing about the report's content, and the design's response is precisely the limitation stated above (Definition 1): intake is a shape check, and storing pointers instead of copies keeps the verification burden on the exporting line.

Chain tip suppression. A holder of the whole chain can present an earlier state as current for an unbounded period, because nothing inside the chain detects a discarded tip. The register hands out a `seal_tip` digest for external anchoring (Proposition 1), but until an anchor exists in a system the chain does not control, the tip is unbound and a suppressed later state is invisible to the register's own verification. The register's `verify_chain` (Proposition 4) confirms internal consistency — every state links to its predecessor, every digest matches — but it cannot detect whether a newer state exists outside the presented segment.

Relation inflation. A submitter can record an agreement, dependency, or incomparability relation that overstates consensus — a single `AGREES` record entered by one party with no genuine counterparty review. The register's projection weights every relation equally; it cannot assess whether a relation was genuinely co-authored or whether the cited reference backing it is accurate. The non-compensatory rule (Proposition 2) narrows the damage: a single block still forces -1, and no volume of agreement lifts it. But agreement inflation can manufacture a misleadingly supportive posture when no block exists — a risk the register cannot close without semantic and institutional authority it does not have.

Projection gaming via declared next use. A caller can choose a narrowly scoped u that avoids known holds — declaring a use that evades the unresolved tension, the outstanding return, or the unreviewed holding that would otherwise cap the posture. The projection evaluates exactly the use submitted; it does not explore adjacent uses or detect that a broader or adjacent use would be blocked (Definition 3). The register's honest response is to report the posture for the declared use with its `state_ref` and reasons intact, so a reviewer comparing the declared use to the state's content can ask whether the scope was chosen to game

the result. The projection is deterministic for identical inputs (Proposition 6), but determinism says nothing about whether the declared use honestly names what is being proposed.

These are instances of a well-documented dynamic, not defects unique to this design. A witness register that certified truth would make these failure modes catastrophic, because a gamed posture would launder bad records into apparent endorsement. The register's design response is to refuse the certifying role entirely: a posture reports what the register contains and what the projection computed, so a gamed +1 overstates nothing but the presence of declared envelopes and the absence of recorded blocks. The attacks also stay inspectable rather than hidden — the stored envelopes, the relation records, the chain tip digest, and the declared next use are the very state a reviewer reads, so a reviewer who asks "does the report ref point at a real report?", "is this the latest tip?", "were these relations genuinely co-authored?", or "does this declared use honestly name what is being proposed?" is asking questions the register state itself exposes. The instrument narrows what gaming can counterfeit; it cannot remove the need for the human judgment those questions require, and it never converts any posture into a safety certification, an accreditation, or a permission.

10 Conclusion

The Witness Register was built to answer one question: can four independent instruments co-exist in a shared record without any of them losing authority over its own vocabulary? The answer is yes, but only because the register refuses to do almost everything a reader might expect of a shared record. It does not rank. It does not average. It does not merge. It does not score. It does not judge; Proposition 2 enforces this in code. It holds envelopes beside one another, records authored relations as separate entries, keeps history append-only and sealed (Proposition 1), and on request emits one bounded posture (Definition 3) carrying the digest of the exact state that earned it and the reasons it holds.

The register's mechanisms are small and each is borrowed from an older tradition — linked timestamping, transparency logs, non-compensatory decision rules, provenance vocabularies, boundary objects. The scholarship section (sec. 5) names what is taken and where each borrowing stops. What is new is not the mechanisms but the refusal: the register assembles them into an instrument that co-registers without aggregation, and the non-compensatory invariants (Proposition 2) are enforced in code, not merely stated in prose. The method section (sec. 6) documents the construction, the formal core (sec. 4) states the register's properties as objects and rules, and three further propositions (sec. 7) establish that chain verification is complete (Proposition 4), return contracts constrain the projection across chains (Proposition 5), and projection is deterministic for identical inputs (Proposition 6) — each verified by the test named in its binding table row. The examples section (sec. 8) exercises the register against real data.

The 3×3 canonical witness battery ships as an executable test that every check rejects its own falsified variant, so a green grid is evidence the checks can fail, not only that they passed. The worked co-registration uses four real envelopes generated by running each line's own public API on its own shipped example, and the honest structure over them — four different subjects — is an incomparability relation and a held posture, not a crown. The return was then met the way the contract said it must be, and completing it lifted exactly the return hold and nothing else.

Black Line remains the positive operating discipline. Golden Line holds the aspirational thread. Red Line is the refusal boundary. White Line marks absence, restraint, and unknowability. The Witness Register stands beside the set, not among them: it has no colour, no substantive question of its own, and no verdict. Its motto is *precedence without information destruction*; *no crown without return*. It lives at `docxology/witness_register`, beside the rest of that index.

11 References

The four line works and their set reader, referenced by name and repository URL in the set's convention (by published convention, never by import or relative path):

- Friedman, D. A. *Red Line*. github.com/docxology/red_line
- Friedman, D. A. *Black Line*. github.com/docxology/black_line
- Friedman, D. A. *Golden Line*. github.com/docxology/golden_line
- Friedman, D. A. *White Line*. github.com/docxology/white_line
- Friedman, D. A. *Line Set*. github.com/docxology/line_set

The scholarly references cited in the design and scholarship sections are carried in `references.bib`, each verified against Crossref, the RFC Editor, or the W3C on 2026-07-29 before use.

The design review this work answers — "The Space Between the Lines" (an external review, 2026-07-29) — is unpublished correspondence and is therefore named here in prose rather than entered as a bibliography item. The repository's `docs/correspondence.md` records, item by item, which of its proposals were implemented verbatim, which were adapted with reasons, and which remain open.

Peter C. Fishburn. Exceptional paper—lexicographic orders, utilities and decision rules: A survey. *Management Science*, 20(11): 1442–1471, 1974. doi: 10.1287/mnsc.20.11.1442. The survey of non-compensatory (lexicographic) decision rules; the projection borrows the shape, not the utility apparatus. Bibliographic record verified against Crossref 2026-07-29.

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. The linked-timestamping scheme the witness-state chain follows: chained digests make record order and content tamper-evident inside the chain. Bibliographic record verified against Crossref 2026-07-29.

Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. RFC 6962, Experimental, June 2013. Public append-only logs and the observer-consistency problem this register's tip-unbound limitation restates. Obsoleted by RFC 9162. Record verified against rfc-editor.org 2026-07-29.

Ben Laurie, Eran Messeri, and Rob Stradling. Certificate transparency version 2.0. RFC 9162, Experimental, 2021. The current CT specification; gossip and external auditing remain outside the log itself. Record verified against rfc-editor.org 2026-07-29.

Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO '89 Proceedings*, Lecture Notes in Computer Science, pages 218–238. Springer New York, 1990. doi: 10.1007/0-387-34805-0_21. The hash-authentication tradition behind digest-over-canonical- content seals. Lecture Notes in Computer Science vol. 435, Springer 1990 (ISBN 9780387973173). Conference held 1989. Publication year from the Springer LNCS volume record.

Luc Moreau and Paolo Missier. PROV-DM: The PROV data model. W3C Recommendation, 30 April 2013, 2013. URL https://www.w3.org/TR/2013/REC-prov-dm-20130430/. Provenance as typed records about artifacts, never edits to them. Status and date verified against w3.org 2026-07-29.

Susan Leigh Star and James R. Griesemer. Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907–39. *Social Studies of Science*, 19(3):387–420, 1989. doi: 10.1177/030631289019003001. Boundary objects and standardized forms; the envelope is built to the standardized-form specification, and Star's caution against overusing the term is adopted. Bibliographic record verified against Crossref 2026-07-29.