RT-2026-08 · ACTIVE

Doctrine as Schema: Toward Machine-Checkable Autonomy Gates for Investigative AI

DRAFT An open research track in progress. It sets out a question we are working on inside a live evidentiary environment, reports what has been measured so far, and names what remains untested. Figures will move as experiments run. We are sharing it to find collaborators, not to present a final conclusion.

Evidence: Observed Partner: Formal methods, or a structured-conditioning ablation

Agent autonomy is usually set by policy prose that no runtime can check. We encode investigative doctrine as a normalized, machine-checkable schema, run a model panel that proposes doctrine for analysts to adjudicate, and specify an autonomy architecture intended to tie permitted autonomy to measured competence. Alongside that substrate we report two negative results: counted coverage is not reachable coverage, and an invariant upheld by convention at many sites is not an enforced invariant.

Updated August 12, 2026

An open research track in progress. It sets out a question we are working on inside a live evidentiary environment, reports what has been measured so far, and names what remains untested. Figures will move as experiments run. We are sharing it to find collaborators, not to present a final conclusion.

Almost every deployed agent system decides what the agent may do on its own in prose. That prose lives in a policy document, a system prompt, or a review checklist, none of which can be evaluated at the moment an action is about to happen. When something goes wrong the investigation is textual: someone reads the policy, reads the transcript, and forms an opinion about whether they match. That is not accountability.

The argument

Investigative work has an advantage most domains lack: it already has doctrine. Intelligence disciplines carry a formal, taught methodology — collection, corroboration, source evaluation, disposition — argued over for

decades. It was not invented for an AI deployment; it is the standard analysts are held to, and the right standard for long-horizon systems.

We encode it as a normalized, foreign-key-linked schema rather than a retrievable document set, so it supports runtime checks and not only search. The doctrine substrate is implemented and operates at doctrine level: resolution is action-conditioned and tiered through a pure decision core; an assignment anchors to a whole crosswalk row with its full methodology lineage; ambient assignment runs unattended, so coverage grows from use and not only authoring; and adjudication is itself a doctrine-emitting event, captured at click time. Doctrine now resolves cumulatively across the objects a case is made of — the tags on entities, on relations and at the document-set level roll up into one view rather than competing for a single winning label.

One invariant runs through all of it: a model suggestion may never override an analyst decision. Machine origin resolves to suggested status at a fixed confidence; analyst origin resolves to accepted at full confidence. We note the second mapping honestly: it encodes the analyst's operational authority as epistemic certainty, and the schema does not yet separate the two — a conflation we regard as a defect to fix, not a feature. Re-typing recommendations stay recommendations. The system annotates; it does not mutate.

That invariant is now doing real work, because doctrine is no longer assigned only by rules and hands. A panel of models proposes the methodology and technique for each relation and entity, choosing from that subject's own taxonomy shortlist, and analysts adjudicate the proposals. The aggregation protocol matters more than the panel: a proposal requires at least two distinct emitters, carries its support ratio and the individual votes rather than a single collapsed label, and **preserves dissent instead of resolving it** — a contested proposal is presented as contested. Two skip conditions encode the sovereignty rule directly in the pipeline rather than in prose. The panel will not touch a subject an analyst has already ruled on, and it will never re-propose something an analyst has dismissed. A machine that cannot argue with a human twice is a cheaper guarantee than a policy document saying it should not.

The autonomy layer sits on top of that substrate, and it is at a different maturity. This is the distinction the title's "toward" is doing work for. Gate semantics are written: absence of a resolved doctrine is not permission; a gate enforcing at accepted must not block on a suggested label. Within the isolated gate component the default is fail-closed — two competence gates are hardcoded not-evaluable, so no model evaluates to an overall pass, by design rather than defect. **System-level fail-closed enforcement is not demonstrated**, because no consumer reads a gate verdict to block anything. An implemented schema underneath an unwired gate is exactly the configuration this track reports on, and it is a more common configuration in deployed systems than the literature suggests.

What we measured

The autonomy layer is specified and not yet wired, so there is no gate pass rate to report and we will not manufacture one. What the substrate has produced instead is a growing body of adjudicated doctrine — and two negative results that we think are the more transferable contribution.

Counted coverage is not reachable coverage. Two independent demonstrations landed in the same window. A scheduled doctrine-assignment task ran **33 consecutive ticks, all of them failed**, on a handler signature error — while its dry-run preview reported healthy, because the preview issues a read and never reaches the conflict arbiter that was raising. The defect has since been fixed and the task has run cleanly ever since; the finding is not the outage but that a coverage metric reported health straight through 33 hard failures. Separately, one rule in a published nine-rule table queried an uppercase flag token the interface never writes. It reported a structural zero, was unreachable for weeks, and was counted as live in telemetry the whole time — and because the rule fed a downstream trust assignment, findings a human had confirmed were being silently recorded as machine suggestions. It has since been corrected. **Work of this kind has to report reachability, not cardinality** — and ours, as built, reports cardinality.

A third observation, smaller but recurring. Two models could not be used in the panel as first specified, and neither failure was about capability. One degraded its structured output under a strict-JSON constraint — malformed keys, placeholder values — apparently because its reasoning-channel output format and the JSON constraint interact badly; we removed it from the panel. Another returned zero groundings until its token budget was raised enough to hold a reasoning phase *and* the JSON payload; at the original budget it spent the entire allowance thinking and emitted nothing parseable. Both present as model incompetence and are serving-parameter artifacts. This is the third such artifact in this programme — an output cap that silently truncated roughly a quarter of one extractor's valid output is the other — and the pattern deserves a name: **before concluding that a model cannot do a structured task, check whether the harness let it finish.**

An invariant upheld by convention at many sites is not an enforced invariant. The sovereignty guarantee is upheld at **eight independent sites**: a resolver status assignment, four conflict-tolerant write paths, an ownership setter, a normalizer preservation path, and a scheduler insert-if-absent contract. No database trigger, no central choke point. All eight are independently breakable, and the predicted failure has already occurred: **one migration broke three of the eight**; two were caught within hours, one went undetected for nearly four days. The honest claim is sovereignty by convention plus idempotency, with a site count and a defect history — more useful than a false claim of structural enforcement.

A first doctrine model is entering training on this substrate, and two pieces of it landed together. The methodology taxonomy is now fully embedded, so a free-text proposal resolves to a taxonomy row by nearest neighbour in vector space rather than by string similarity — with the string matcher kept as a fallback behind the same contract, so the two paths cannot drift apart. And the doctrine source corpus has been digitized into supervision rather than into text.

The second part is worth describing precisely, because it is the training signal and it is not what "we have the doctrine documents" would imply. Each source document is reduced to a canonical text in which per-figure model transcriptions are interlaced with the best available native extraction. Doctrine and methodology publications turn out to be diagram- and layout-dense enough that native text alone loses the substance — a property of this literature we did not anticipate, and one we would flag to anyone else planning to train on it. Against that text a model proposes methodology and technique groundings, and each grounding carries a **verbatim span mapped back to a true character offset and page in the source document**, together with the model's reasoning, its confidence, the candidate taxonomy rows it considered, and the row it resolved to.

The supervised unit is therefore *span in doctrine → row in taxonomy* — anchored, and checkable against the page it came from — rather than a document-level label. The same pass emits a catalogue of where the corpus does **not** ground — and that catalogue turned out to be the most useful thing here, for reasons we did not intend.

At that point the argument of this track becomes testable rather than architectural: doctrine as *structured conditioning* for a model, measured against an undoctrined baseline. We have no result to report yet and will not preview one.

The coverage catalogue was three-quarters wrong, and it was wrong in exactly the way this track argues coverage metrics fail.

Its first definition was: a gap is a machine-grounded method with no corresponding note file in the doctrine vault. Reasonable, and false. Analysts record a methodology in either of two places — an inline link in a note's body, or a structured field in its header — and they deliberately leave the linked note *file* uncreated, because stub creation is queued for automation. A dangling link is coverage, not absence. Measured against file existence, **74% of the flagged gaps were false positives**: the method was documented, only its stub was missing. The remaining quarter are real — machine groundings that appear nowhere in the doctrine at all.

A second finding explains why no single-channel measurement could have caught this. The recording convention drifted as the corpus grew: every note carries inline links, but only the earlier ones also carry the structured header field, and that habit fades. Coverage is the *union* of the two channels, so any metric reading one of them is biased by document age. We would flag that to anyone mining an expert-maintained corpus — the humans' convention moves, and a measurement built against the current convention silently misreads the older material.

The redesign resolves both the human references and the machine groundings to taxonomy rows before comparing them, which drops non-method links out naturally, and splits the output three ways: real gaps, deferred stubs, and the artifact we actually want — the **overlap and disagreement between what the expert documented and what the machine grounded**. Machine-only groundings are either real gaps or machine over-reach, and separating those is analyst work. Expert-only references are machine recall misses, and we cannot count them honestly yet, because the reference links point at documents, disciplines and sections as well as methods; the raw number is noise until that anchoring lands.

One detail matters more than the correction. **An analyst reading the output caught this, not the pipeline.** A catalogue that counted files reported a coverage figure that was three-quarters wrong, every count in it was accurate, and no telemetry we had would have flagged it — because the counts were never the problem, the definition was. That is the argument of this track arriving in our own work, and it is why we publish the negative results rather than the feature inventory.

Doctrine corpus cardinality is shared under NDA. Coverage telemetry emits raw totals with no ratio and no history — itself part of the first negative result.

What this does not show

No autonomy has yet been gated by doctrine. The central integration — a competence gate that reads the doctrine schema — is specified and unbuilt, and the gate module currently contains no doctrine references at all. Nothing here demonstrates a graduated unlock, and the thresholds that would define one are still undefined. Gate verdicts are advisory: nothing consumes them to block a promotion, so fail-closed today is a property of an isolated component rather than of the running system. The title says *toward* for this reason.

Doctrine is also unversioned. No doctrine table carries a validity interval or a superseded-by pointer, so a past decision cannot be replayed against the doctrine that was in force when it was made. For a governance argument that is the gap we would fix first, and it is squarely on the roadmap rather than in the too-hard pile.

Inter-annotator agreement is not yet computable, and the reason is the same phenomenon the rest of this track describes. More than one analyst has adjudicated, but until recently no per-analyst identity was stamped on a doctrine assignment, flag, edge adjudication or integrity finding — every human action resolved to one generic identity in both stores. Authoring identity is now captured on the doctrine path and is being threaded through the remaining mutation paths. A governance schema that records *that* a human decided but not *which* human cannot support an accountability claim, and this was the fourth instance in this programme of a system counting something it could not attribute. It is the instance we are closing.

One caveat we would raise ourselves: agreement on this task may turn out to be uninformative. Adjudicating a machine proposal against a resolved matter is closer to verification than to annotation, and where the analyst is confirming an outcome they already know, disagreement measures transcription error rather than rubric ambiguity. Typing the *mechanism* of a failure is the axis where we would expect real variance, and it is the axis we intend to measure first.

A reproducibility tension remains unresolved. The enforcement threshold catalogue and the gate definitions live deliberately in a local-only store. An autonomy-gates argument whose thresholds cannot be published has an artifact problem, and we have not decided what to ship.

Documentation drift closes the list: our own canonical doctrine specification is stale relative to the loaded corpus — the first document a reviewer would open.

Relationship to prior work

Runtime governance for agents is an active area and we are not claiming the category. Specification-driven runtime enforcement, machine-readable governance artifacts attached to a deployed agent, and policy-as-code interception at execution boundaries are all being built and published; the shared premise is that policy has to be evaluable at the moment of action rather than reconstructed afterwards from a transcript. We agree with that premise and this work rests on it.

Two things here are less common. The first is the source of the policy: investigative doctrine is a pre-existing professional standard with decades of argument behind it, not a rule set authored for the deployment, which

changes what "compliant" means and who gets to adjudicate it. The second is the reporting stance. The literature is largely about enforcement mechanisms that work; our contribution is two measurements of enforcement that *silently did not* — and the observation that neither would be caught by any coverage metric currently in use, including our own.

- Wang et al., *AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents* — ICSE 2026 (arXiv:2503.18666)
- Mavračić, *Policy Cards: Machine-Readable Runtime Governance for Autonomous AI Agents* — arXiv:2510.24383
- *Runtime Governance for AI Agents: Policies on Paths* — arXiv:2603.16586

Where this is going

Two tracks, and they are independent enough that a partner can take either.

On the governance side: add a per-rule reachability check as standing telemetry, so the second failure is caught by a test rather than by someone reading code. Persist coverage as ratios with denominators and history. Run the sovereignty ablation — inject a suggestion at each of the eight enforcement sites and record which fail open. Version the doctrine, then wire the first gate that reads it.

On the modelling side: the training run now beginning is the first opportunity to test the claim under the title. Doctrine as structured conditioning, measured against an undoctrined baseline on long-horizon reasoning quality, is an ablation we have not found in the published literature, on a taxonomy of this size, and the substrate for it is live rather than planned.

About RedTorch

RedTorch, Inc.® is an investigative intelligence and applied AI firm, operating continuously since 2016 from Inglewood, California under California Bureau of Security and Investigative Services private investigator license 189236. Two practices run in parallel — investigative services (due diligence, litigation support, digital forensics, fraud investigation, asset recovery) and AI and strategy consulting — across more than twenty-five service lines. Casework runs on a platform we built, on GPU infrastructure we own. Privileged material never leaves our custody, and provenance, sanitisation and regulatory mapping are built into the platform's data model and operating workflows rather than added as a reporting layer.

RedTorch's platform is a closed-loop environment for evaluating, improving, and governing models on real evidentiary work. Evidence is captured with its lineage intact; machine claims are checked against their sources; analyst acceptances *and* rejections are preserved as supervision; the case graph rather than embedding similarity decides what a model reads; competence is measured from production exhaust; and the autonomy a model is granted is gated on that measurement. Each track in this collection documents one stage of that loop. We are building for systems that reason today and act tomorrow — a harder standard than benchmark performance, and a more useful one.

The corpus is what makes the research possible. It combines real privileged investigative text, an industry taxonomy, model-generated claims at scale, and a growing adjudication layer — analyst acceptances, dismissals, ruled-out hypotheses, doctrine assignments and integrity findings. Coverage and attribution vary by signal, and several of those gaps are themselves subjects of these tracks. What cannot be assembled after the fact is the pairing: a decade of privileged casework alongside the machine output produced against it and the expert judgement passed on that output. Corpus metrics are shared under NDA. This track works on the methodology layer: the professional doctrine analysts are held to, encoded rather than paraphrased.

Collaborate

The unrun experiment is fault injection across the eight sovereignty sites, paired with a gate-threshold sensitivity sweep against pre-specified unlock criteria — no training compute required, and it would carry this track's strongest empirical section. RedTorch, Inc.® supplies the doctrine schema, an adjudication fabric with click-time capture, competence exhaust from a deployed fleet, and a documented defect history. We lack a formal-methods partner and the compute to calibrate autonomy thresholds. Researchers and labs interested in this work can get in touch.