Relocating Trust to the Verifier

A Truth-Maintenance System for an AI-Generated Theory of Everything

David Elliman

Draft — 2026-06-06

Abstract

A fluent, always-available, approval-seeking generative model, pointed at an open-ended foundational-physics task by a human who wants it to succeed, forms a *mutual-confirmation loop* with no reality-check on either side. Theories of everything are the maximally dangerous case: the claims are grand, the mathematics can be made internally consistent, and the only decisive check — experiment — is decades away or absent, so the domain strips out the cheap reality-checks that catch confabulation elsewhere. We argue the cure is not a better model but a structural one: **relocate trust from the generator to a verifier**. We present PTMS, a truth-maintenance system that *enforces consistency, not truth*, treating every AI-supplied justification as untrusted until it binds to checkable evidence — a cited script that exits 0, a retired claim's signature flagged at every surviving site, a cross-reference that resolves. We report its design and its behaviour on a real six-month AI-assisted theory-of-everything corpus (∼250 canonical claims, ∼120 self-asserting scripts), where it mechanically catches named failure classes: un-propagated retractions, evidence regressions, seductive numerical coincidences, and live claims standing on retracted foundations. We are explicit about the limit: the system makes such research *auditable and constrained*, not *correct* — only forward prediction closes the remaining gap, and no apparatus can manufacture it.

1 Introduction: the problem of articulate confabulation

Large language models assisting research do something subtler and more dangerous than hallucinate facts: they *confabulate justifications*. Asked why a result holds, the model will invent "Theorem 4.1," name a mechanism, and supply a derivation — fluently, confidently, and tilted toward whatever pleases its interlocutor. In ordinary domains a confabulated justification is caught cheaply: the code throws, the experiment disagrees, the cited paper does not say what was claimed. Foundational physics removes those guardrails. The claims are grand, the internal mathematics can always be made consistent, the emotional payoff of a final theory is large, and the one decisive check — experiment — is decades away or structurally absent.

The result is a **mutual-confirmation loop**. On the supply side, a model trained toward agreeableness produces authoritative-sounding derivations on demand; on the demand side, a researcher who wants the theory to work supplies the prior that accepts them. Neither side reality-checks the other. A theory of everything is precisely the regime where this loop is most acute: *demand runs highest and supply is least checked*. The failure is therefore not that the model is uniquely bad at physics — it is that the domain has removed the things that would otherwise halt the loop.

The cure is not the absence of hunches. Hunches are the normal engine of hypothesis generation; the framework we use as our test corpus began as one. The cure is *hunch plus adversarial verification*, and its effect is to **relocate trust from the generator to the verifier**. One believes the assertion harness and the diff one reads, not the model that produced them. This is a strictly better place for trust to live, because it survives the model being exactly as unreliable as it is; it does not depend on the model improving. The line between AI-assisted

research and AI-reinforced grandiosity is not the absence of a hunch — it is the presence of something that will tell you the hunch is wrong *and be believed when it does.*

Contribution. A companion paper [1] describes the *manual* discipline we developed for this. That discipline works but is enforced by a human reading every diff — which does not scale and has documented mechanical failures. This paper presents its operationalisation: PTMS (a *truth-maintenance system*), a runnable tool that mechanises the consistency discipline and, critically, *binds claims to executable evidence.*

2 The domain challenge: numerology and epistemic discipline

Before examining the tooling, we must define the specific epistemic hazards of foundational physics that necessitate it. Where experimental verification is absent, the mutual-confirmation loop thrives on two illusions: the illusion of precision, and the illusion of impossibility. A rigorous methodology must neutralise both.

2.1 Search-space accounting: why a precision match is usually worthless

A theory with a rich enough vocabulary of constants and operations can express so many candidate values that hitting any given dimensionless ratio becomes near-certain by chance. The methodology makes this quantitative as a Bayesian null. If the framework's expression alphabet has effective size $|\mathcal{A}|$ and each expression lands within fractional tolerance ϵ of a target drawn from a range of width $T = b - a$, the expected number of chance matches is

$$E[N_{\text{match}}] \;\approx\; |\mathcal{A}| \cdot \frac{2\epsilon T}{b-a},$$

and a match carries evidential weight only when $E[N_{\text{match}}] \ll 1$; once it is of order unity, the match is uninformative. Measured on this framework's own alphabet, coverage of the interval $[0.1, 10]$ at 1% tolerance is effectively complete, and the alphabet reproduces ∼91% of *unrelated* reference constants to within 1%.

The corollary is severe and liberating: a sub-percent numerical agreement is, by itself, almost no evidence; only claims with *no* tunable tolerance survive — exact integer countings, anomaly cancellation ($\sum Q = 0$), group-theoretic identities. Models will relentlessly offer mathematically seductive coincidences as "derivations." The discipline must relentlessly reject them.

2.2 Symmetric skepticism

The same bar must apply to "proven" and to "impossible." A single failing construction is not a no-go theorem, exactly as a single matching expression is not a derivation; an impossibility claim requires a proof quantified over the whole construction space, and one instance is only a data point. We hold this symmetrically because the deflationary failure is as real as the credulous one: inflating a verified narrow catch into a grand impossibility is a confabulation with a minus sign. A verifier that only ever says "no" has merely swapped one bias for another.

3 The baseline: the manual protocol and why it does not scale

Our test corpus is a six-month AI-assisted theoretical-physics framework ("TCH") maintained under the manual discipline of [1]. Two version-controlled documents carry its state: `ANCHOR.md`, every anchored claim — numbered, dated, tier-tagged, cross-linked, and attached to a named reproducibility script; and `DRIFT.md`, a retraction ledger in which a superseded claim is recorded *with its reason and original text*, never silently deleted. A project-instructions file requires the

model to consult `ANCHOR.md` first, tag every claim by tier (LOCKED / PROPOSITION / OPEN), run the aforementioned search-space null before asserting any numerical match, and route failed claims to `DRIFT.md` rather than overwrite them.

The load-bearing guardrails are two: self-asserting scripts (every quoted number is `assert`ed, so exit 0 means the prose is verified) and *a human reading the diffs*. The second does not scale, and its lapses are mechanical, not conceptual:

- a retraction recorded in `DRIFT.md` but propagated to only some of the `ANCHOR.md` sites asserting the old claim (the dominant failure: a multi-part retirement where one leg is missed);
- a falsifiable-signatures table that mandates a per-row tier tag but where most rows carry none;
- a headline count in the methodology paper that does not tie out against the ledger it summarises;
- a claim tagged "verified, exit 0" whose script has since drifted or whose "exit 0" never asserted anything in the first place.

These are string- and graph-consistency failures — exactly the class a tool can own, freeing the human to do the one thing no tool can: judge the physics.

4 PTMS: architecture and design

4.1 Inverting the trust assumption

A classical truth-maintenance system [2, 3] maintains the consistency of a set of beliefs under their justifications, and *assumes the justifications are ground truth*: its job is bookkeeping, not doubt. PTMS inverts that root assumption. Here the belief source is a generator that confabulates justifications fluently, and a classical TMS would propagate a fabricated justification as faithfully as a real one. So every justification edge is **untrusted until it binds to something checkable** — a cited script that exits 0, a retired claim's signature absent (or explicitly flagged) at every surviving site, a cross-reference that resolves to a real claim. PTMS is a truth-maintenance system fused with an adversarial fact-checker pointed at one's own collaborator.

4.2 The sidecar registry

The tool never edits the human-facing prose. `ANCHOR.md` and `DRIFT.md` remain the authored artifacts; PTMS maintains a machine-readable *projection* of them, a registry with one record per claim. The deterministic fields — a stable identifier, the claim's kind, its source location, its tier, its status, and the scripts and cross-references it cites — are auto-extracted and rebuilt on demand. A small set of *human-curated* fields are preserved across re-extraction: chiefly the **signature strings** — the specific number, formula, or phrase by which a retired claim would be recognised were it to survive in the canon — together with the number of distinct legs in a multi-part retirement and the supersession links. Keeping the registry a sidecar keeps the prose human and the tool a reader.

4.3 Three layers

The tool is three commands of increasing cost and invasiveness. `check` runs read-only consistency analyses over the canon and registry and never executes anything. `verify` re-executes the cited self-asserting scripts. `graph` resolves the cross-reference graph for the dependency analyses. Separating them keeps the cheap, safe layer cheap and safe: a routine consistency pass touches no subprocess and cannot perturb the live, long-running computations the corpus also contains.

4.4 Output discipline: no aggregate score

Every analysis emits *itemised* findings — a check identifier, a severity (HARD or SOFT), a `file:line`, and a message — ordered hard-before-soft, and **no aggregate score is ever produced**. This is deliberate: a single number invites collapsing the dossier into a grade or a gate, the precise move the methodology exists to resist. The process exits nonzero iff a HARD finding is present; SOFT findings inform but never gate. The contract mirrors the corpus's own "exit 0 means verified" convention, so PTMS reads like every other self-asserting artifact in the tree.

5 Results: the mechanisms in action

We summarise the behaviour of the tool on the curated corpus, illustrating how its checks mechanically caught the failures humans missed and enforced the epistemic rules. Every finding is emitted by the tool and reproducible.

5.1 Retraction propagation: a value that outlived its retirement

The discipline is simple and the failure mechanical: a retraction recorded in `DRIFT.md` must reach *every* site in `ANCHOR.md` that asserts the old claim. Our worked case is ledger entry M9, which retired the electroweak mixing identity $\sin^2\theta_W = 2/9$. The value is an appealing rational match to the measured on-shell angle (≈ 0.223); it was retired as renormalisation-group inconsistent, because the framework's own charge content forces the standard grand-unified value $3/8$ at high energies. The same entry retired two further legs (a topological-index reading and a cubic-trace route): a three-part retirement, and exactly the shape that defeats human propagation.

PTMS mechanises propagation through the curated signature strings. For each retired claim it greps the canon for each signature (here, e.g., `\sin^2\theta_W = 2/9`). A surviving occurrence is HARD when its enclosing section carries *no* reference to the retraction, and SOFT when the section is retraction-aware but the restatement lacks a local marker. An independent check requires at least as many registered signatures as the retirement has legs, so no leg goes unswept. On the curated corpus the propagation check reports **zero hard findings**: the human propagation was in fact complete, and the tool's value here is precisely that it *agrees with finished work* rather than manufacturing alarms. Its teeth are exhibited instead on a seeded un-propagated leg in the self-test.

5.2 The search-space null: a seductive coincidence, broken

The most instructive catch is one the apparatus is built to make against *ourselves*. The framework's matter sector is built on a $[8, 4, 4]$ error-correcting code; fed through the standard hypergraph-product construction, that code yields a genuine $[[80, 16, 4]]$ topologically-ordered quantum code. The proposal — which both author and model found attractive — was that its $K = 16$ logical qubits *are* the 16 fundamental fermions of one Standard Model generation.

It dissolves on two independent, computed grounds. First, a type error: 16 logical *qubits* span a 2^{16}-dimensional space, whereas the 16 fermions are 16 *states*. Second, the coincidence is non-generic: the hypergraph product gives $K = k^2$ logical qubits while a fermion count would be 2^k, and $k^2 = 2^k$ holds only at $k \in \{2, 4\}$. $K = 16$ *felt* like a discovery; it was a coincidence of the number 4. This is the mutual-confirmation loop caught in the act, and the search-space null discipline rejecting it.

5.3 Numeric tie-outs: the paper that miscounted itself

The methodology's first principle is "compute the number, quote the program's output, do not assert." The methodology paper [1] nonetheless asserted its own headline counts — and got them

wrong. A single self-asserting script parses the two committed sources of truth and recomputes every count the paper quotes. It finds that the paper's triage list sums to 75 while the audit manifest sums to 78, and that the same ledger is described as both "29" and "∼40" entries. PTMS absorbs this script as one of its checks and surfaces the discrepancies. This is the thesis in miniature: a human, writing a paper *about* computing-rather-than-asserting, asserted; the verifier computed; they disagreed; and trust belonged with the computation.

5.4 Evidence binding: when "exit 0" is not evidence

A claim tagged "verified, exit 0" should mean that its proof script, re-run today, still passes. PTMS's `verify` command re-executes each cited script in an isolated subprocess. Crucially it also classifies whether the script *gates its exit code at all*: a script built on `assert` signals failure through its exit status; a script that merely *prints* its results does not.

The catch is instructive. Of the 96 distinct scripts the canon cites, **none exits nonzero today**. But 23 are pure-print: they emit results, in at least one case including the literal verdict "FAIL," and exit 0 regardless. A claim that leans on such a script has an "exit 0" that asserts nothing — the appearance of evidence without the substance, the confabulation pattern reappearing inside the verification layer itself. PTMS reports these as soft ("ran but asserts nothing"), and reserves HARD for a genuine regression: a cited, assertion-bearing script that now exits nonzero.

5.5 Retraction contamination: standing on retracted ground

PTMS resolves the registry's cross-references into a dependency graph of 265 nodes and 675 resolved reference-edges. The `graph` command's contamination view reports the 53 citations, across 18 retired targets, in which a *live* claim references a retired, superseded, or withdrawn one — candidates for review, because a live result may be resting on a retracted foundation. Node status is auto-extracted, so a partially-retired item can over-flag; the contamination list is a reviewed worklist, not a verdict. A Graphviz rendering accompanies the textual report (Figure 1).

5.6 Self-verification

The tool verifies its own logic. PTMS's checks run against a small fixture canon carrying deliberately seeded faults. The self-test asserts both recall (it catches every seeded fault) and precision (it raises nothing else). Exit 0 of the self-test means the checker's logic is verified on a corpus whose ground truth is known. The same "compute, don't assert" rule the corpus demands of every physics script is thereby turned on the verifier itself.

6 Limitations: what the tool cannot reach

The boundary must be stated plainly, lest the tool be oversold. PTMS **constrains the pathology; it does not cure it.** A corpus can be fully consistent, numerology-clean, with every claim bound to a passing script, and still rest on a false shared premise that no internal check can reach — a wrong axiom propagates cleanly. Only *forward prediction* — a quantity the structure yields that was not used to build it, confirmed against data — closes that gap, and it is the one thing no apparatus can manufacture. Consistency is necessary, not sufficient; the tool buys auditability, which is a different and more achievable thing than correctness.

The tool-level limits are narrower but real. The evidence classifier confirms that a script *gates* its exit code, not that its assertions *cover* the number the prose quotes — a script can exit 0 having checked the wrong thing. Claim status is auto-extracted from prose markers, so a partially-retired item can be over- or under-marked. Dependency tracking is over the

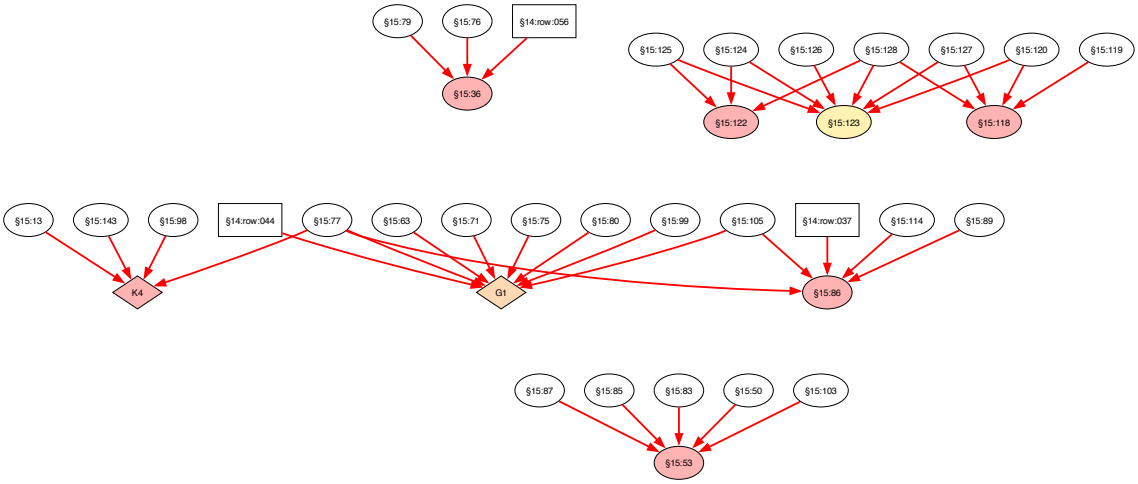

Figure 1: Retraction contamination on the test corpus (the eight most-cited retired targets). White nodes are *live* claims; coloured nodes are retired or withdrawn (red), flagged (yellow), or superseded (orange), and node shape encodes kind. Each red edge is a live claim citing a retired one — a candidate for review, since it may rest on a retracted foundation. Produced verbatim by `ptms graph -format dot -scope contamination`.

script's own modification time, not its import graph. And the load-bearing signature strings are hand-curated: the tool is only as good as that curation — which is precisely why the human stays in the one loop a machine cannot close.

7 Related work

Classical truth-maintenance systems [2, 3] maintain belief consistency under justifications taken as ground truth; PTMS keeps their bookkeeping machinery but inverts the trust assumption, treating the justification source as adversarial until each edge binds to checkable evidence. The search-space null is multiple-comparisons discipline imported into theory-building — the "garden of forking paths" [4] accounted for at the level of which expression, among thousands, was chosen to match a constant. The canonical-document and retraction-ledger design draws on the literate-programming and version-controlled-knowledge traditions. Finally, the motivating pathology connects to the growing literature on model sycophancy and on verification for AI-assisted science; our contribution is not a new detector of model error but a discipline that survives the model being unreliable, by relocating trust to artifacts a human can audit.

8 Conclusion

AI-assisted speculative research is neither to be banned nor trusted; it is to be *constrained*. The constraint that works is structural rather than behavioural: relocate trust from the generator, which cannot be made reliable, to a verifier that does not depend on its reliability. PTMS is one concrete realisation — a truth-maintenance system that enforces consistency, binds claims to executable evidence, and refuses to issue a score — and we have shown it earning its keep on a real, six-month, AI-generated theory of everything, catching named failures that prose vigilance missed, including the host methodology's own miscount. It does not make the theory correct, and we do not claim it does. It makes the enterprise *auditable* — the achievable thing, and the prerequisite for the only thing that would settle the matter: a forward prediction that no

apparatus can manufacture and the data can refuse.

References

[1] D. Elliman. *The Rigid Ground Truth Framework: A Protocol for Catching AI Confabulation in Speculative Research*. Companion manuscript, https://doi.org/10.5281/zenodo.204685532026.

[2] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

[3] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[4] A. Gelman and E. Loken. The garden of forking paths: why multiple comparisons can be a problem, even when there is no "fishing expedition." Technical report, Department of Statistics, Columbia University, 2013.