The Epistemology Is the Compiler

Build-gate enforcement of evidential standards in an evidence-graded historical knowledge graph

DeepOntology Collective & DeepEpisteme Research Group
deeptunisia.org · deepontology.org
Correspondence: contact@deeptunisia.org

ABSTRACT

Knowledge bases documenting contested political history face an institutional breakdown: their evidentiary standards are enforced socially, as editorial conventions, review guidelines, or post-hoc fact-checking. Social enforcement degrades under volume, fatigue, and political pressure, and collapses when cheap language models generate plausible synthetic citations. We present DeepEpisteme, an open-source epistemic compilation framework that converts evidential policies into machine-enforced build-time invariants. We evaluate DeepEpisteme against two reference corpora: Ruritania, a controlled synthetic jurisdiction used for deterministic regression testing, and DeepTunisia, a real-world Mediterranean proving ground documenting political, security, and economic networks (1956–2026). Every claim-bearing record in the graph composes a nine-field evidential interface. The compiler enforces strict admissibility constraints: an inferred claim lacking an explicit falsifier aborts the build, a low-confidence claim lacking a named claimant aborts the build, and an impossible date or contradictory temporal interval halts publication. Uncertain spans resolve into four-field fuzzy intervals whose clamping is published as open data, never resolved by silent arithmetic. An adversarial self-audit of the shipped implementation surfaced seven real bypass vulnerabilities, including silently stripped fields, calendar rollover, and inverted intervals. Each bypass was closed with dedicated validators and regression fixtures. A mutation testing campaign against the validator layer detected and killed 25 of 26 deliberate breakages (96%), providing empirical evidence that the test suite detects tested classes of deliberate implementation weakening while exposing the blind spot of testing solely against clean canonical data. The compiler is paired with an editorial review workstation and Agora, an optional downstream deliberation module whose proposals re-enter through the compiler gate. DeepEpisteme demonstrates that the evidentiary discipline long envisioned for semantic knowledge bases can be compiled deterministically. Violation of the declared evidential contract is structurally prevented at compilation. The dataset cannot be quietly wrong with respect to the implemented evidential contract.

Keywords: knowledge graphs · evidence grading · provenance · build-time validation · epistemic basis · fuzzy temporal intervals · computational social science · verifiability

1. INTRODUCTION

Knowledge systems that track contested political history routinely suffer from an evidentiary collapse:

source → interpretation → inference → assertion

Raw archival sources, interpretations, inferential leaps, and assertions are flattened into a single, undifferentiated fact. Once an assertion enters a public database or reference graph, its inferential scaffolding disappears. Readers cannot distinguish an official decree from an investigative inference or a circulating rumor. In sensitive historical domains, this collapse invites narrative capture, political laundering, and editorial drift.

The conventional response to this problem is social policy. Projects write editorial guidelines, establish peer review panels, or conduct post-hoc fact-checking [11, 24]. In practice, social enforcement fails. Guidelines are forgotten during bulk ingestion. Review panels face backlogs. Post-hoc corrections arrive long after a false assertion has propagated through secondary literature. Furthermore, the cost of generating plausible synthetic documents has dropped to near zero. Language models fabricate legal precedents that reach court filings [23] and generate persuasive text at scale [20, 21]. A knowledge base that relies on human memory to police evidential obligations cannot survive adversarial pressure.

This paper presents **DeepEpisteme**, a framework that treats **the evidence policy as the compiler**. In DeepEpisteme, plain-text YAML files in version control are the canonical dataset. The build pipeline is a deterministic compiler that parses, validates, resolves, derives, and emits the knowledge graph. If a record violates an evidential invariant, compilation aborts. No dataset is emitted. An inference without a testable falsifier does not enter the graph. A low-confidence assertion that omits its claimant does not enter the graph. An impossible calendar date does not enter the graph.

The architectural foundation of DeepEpisteme rests on a single division of responsibility:

> **Axiom A1.** The compiler verifies that a claim conforms to its declared epistemic contract. It does not determine whether the underlying epistemic judgment is correct.

Axiom A1 defines the boundary between machine verification and human judgment. Whether an archival

document proves an appointment is a substantive question for human researchers. Whether a claim declared as `inferred` carries the reasoning and falsifier required by its schema is a formal question for the compiler. DeepEpisteme does not automate historical truth. It automates the enforcement of an epistemic constitution, refusing to compile any record that violates its declared obligations.

We evaluate DeepEpisteme across two reference environments:

1. **Ruritania**: A controlled synthetic jurisdiction (~25 records, 5 sources, 32 assertions) designed for fast onboarding and deterministic regression testing.
2. **DeepTunisia**: An adversarial Mediterranean proving ground tracking political, military, security, and economic networks across four regime ruptures (1956, 1987, 2011, 2021). DeepTunisia is not the contribution being generalized; it is the adversarial real-world corpus through which the contribution is currently exercised. At the canonical dataset snapshot frozen for this release (September 2026), DeepTunisia holds 1229 sources, 444 people, 422 positions, 362 relationships, and 100 events.

Throughout this paper, we use the phrase **"the dataset cannot be quietly wrong"** with a precise technical meaning: the dataset cannot be quietly wrong with respect to the implemented evidential contract. It does not mean the underlying historical sources are infallible. It means an epistemic status degradation or schema-level violation of the declared evidential contract cannot slip into the published graph undetected by the build gate. A weak claim cannot quietly pose as a primary fact.

Contributions

This paper makes four contributions:

1. **An Executable Claim Contract (§3)**: A nine-field evidential interface composed uniformly across all claim-bearing entities. It couples a two-axis grading model (confidence A–D and verification state) to a deterministic derivation mapping (`deriveBasis`), a four-field fuzzy temporal interval algebra, and explicit derivation provenance.
2. **A Verified Invariant Compiler (§4–§6)**: A deterministic build engine executing 24 formal validators (V1–V24). We document an adversarial self-audit that exposed seven real bypass vulnerabilities in our initial implementation, alongside a mutation testing campaign that killed 25 of 26 deliberate breakages (96%), providing empirical evidence of detecting tested classes of implementation weakening and demonstrating the necessity of synthetic negative fixtures.
3. **A Research-to-Engineering Feedback Methodology (§7)**: A specified blinded inter-rater evaluation protocol and multi-model benchmark design to translate recurring evidentiary disagreements into formal compiler invariants via a 12-category taxonomy.
4. **Honest Self-Measurement as Architecture (§6.4, §9.5)**: A build system that programmatically rewrites its own published statistics directly from the graph, preventing narrative prose from drifting from underlying data. We specify the complete architectural decoupling of the jurisdiction-agnostic compiler (Layer 1) from local jurisdictional ontologies (Layer 2) and canonical data (Layer 3).

---

2. BACKGROUND AND EVIDENTIAL THREAT MODEL

2.1 The Contested Knowledge Problem

Historical knowledge graphs that document state power, corruption, and elite networks operate in a hostile evidentiary environment:

- **Living Subjects**: Individuals documented in the graph often hold active political or military office. False claims carry direct legal and personal hazards.
- **Asymmetric Sources**: Official gazettes provide exact legal dates. Investigative journalism reports informal networks. Rumors circulate in partisan press. Collapsing these distinct tiers into uniform graph edges launders rumor into official decree.
- **Synthetic Fabrication**: Low-cost generative models produce counterfeit citations, non-existent decrees, and fabricated news stories that mimic genuine journalistic prose.
- **Narrative Drift**: Authors frequently alter confidence grades or omit cautionary notes to fit preferred political narratives.

2.2 Epistemic Manipulation Patterns

We formalize seven specific threat patterns that target contested knowledge bases:

Threat Pattern	Mechanism	Compiler Defense
1. Provenance Laundering	Converting circulating rumors into documented historical facts.	**Axiom A1 & Basis Derivation**: Basis is derived from source tiers; Grade A requires official primary documentation.
2. Confidence Laundering	Elevating a weak claim to Grade B without corroborating evidence.	**Proposed Validator V25**: Grade B claims must cite ≥ *2* sources or explicitly declare `attributed_to` (§7.4).
3. Inference Laundering	Presenting an analytical deduction as an observed historical event.	**Validator V18**: `basis: inferred` requires non-empty `reasoning` and testable `falsifiable_by` fields.
4. Source Multiplication	Treating syndicated reprints of a single wire story as multiple corroborating sources.	**Source Independence Protocol**: Requires publisher lineage tracking in review records.
5. Temporal Laundering	Imposing false precision by assigning an exact date to an appointment known only by year.	**Fuzzy Interval Algebra**: 4-field intervals reject exact days unless supported by calendar text.
6. Editorial Overwrite	Silently modifying canonical records during community discussion.	**Quarantined Proposal Pipeline**: Deliberation cannot mutate `data/*.yaml`. Changes must pass the outbox gate.
7. Agent Escalation	Autonomous research agents committing unvalidated candidate records directly to data.	**Strict Outbox Isolation**: Agents write candidates strictly to `data/contrib/`. Compilation requires human sign-off.

2.3 Design Principles

DeepEpisteme enforces eight foundational principles:

- **P1: Universal Envelope**: Every claim-bearing record composes the shared evidential interface (§3.1).
- **P2: Git as Canonical Store**: Plain-text YAML under version control is the single source of truth; the build is the gatekeeper (§4.1).
- **P3: Graph as Open Data**: Every compiled layer ships as downloadable, byte-reproducible JSON, CSV, and GeoJSON (§4.3).
- **P4: Machines Propose, Humans Dispose**: Autonomous agents write candidate proposals to an outbox; human editors commit changes (§4.3).
- **P5: No Inference Renders as Fact**: Inferred edges render with distinctive visual styling and explicit computed markers (§3.5).
- **P6: Additive Evolution**: Schemas evolve additively; existing valid records remain valid without forced mass rewrites.
- **P7: Numbers Are Claims**: Quantitative values carry explicit dates and source provenance (§3.3).
- **P8: Direction and Time on Every Edge**: Typed relationships mandate explicit temporal intervals and direction semantics (§3.4).

2.4 The Three-Tier Epistemic Hierarchy

To avoid confusing mechanical validation with historical truth, DeepEpisteme partitions epistemic responsibility across three distinct tiers:

- **Tier 1: Mechanical Compliance (The Compiler)**. The compiler deterministically verifies envelope syntax, required fields, date validity, interval consistency, and basis derivation rules. It structurally prevents undeclared evidential degradation.
- **Tier 2: Human Epistemic Judgment (Scholarship)**. Human researchers and domain experts evaluate whether archival sources justify confidence grades, whether inferential reasoning holds, and whether historical assertions are credible.
- **Tier 3: Social and Governance Resistance to Gaming (Community and Oversight)**. Peer review, public challenge periods, git history transparency, and quarantined outbox moderation prevent actors from gaming declarations (such as falsely marking an uncorroborated claim as Grade B to evade inference obligations).

Crucially, the compiler does not eliminate epistemic gaming; it makes certain forms of gaming explicit, typed, reviewable, and version-controlled, transforming the attack surface from invisible metadata corruption into an explicit, version-controlled human governance problem.

---

3. DEEPEPISTEME ARCHITECTURE AND COMPILATION PIPELINE

3.1 Overview and Pipeline Flow

The compilation pipeline (`scripts/build-data.ts`) transforms declarative YAML into verified graph artifacts through a deterministic sequence:

```
data/*.yaml ⟶ Strict Parse ⟶ Interval Resolution ⟶ Derivation ⟶ Validator Gate ⟶ Emission
                                   (V19)                (V21, V22)          (deriveBasis)          (V1-V24)          (JSON/CSV)
                                    |
                                    [V25 proposed extension candidate]
```

Compilation runs in a single process. Errors accumulate in an `errors[]` array. If any hard invariant fails, compilation aborts with a non-zero exit code. No dataset is emitted.

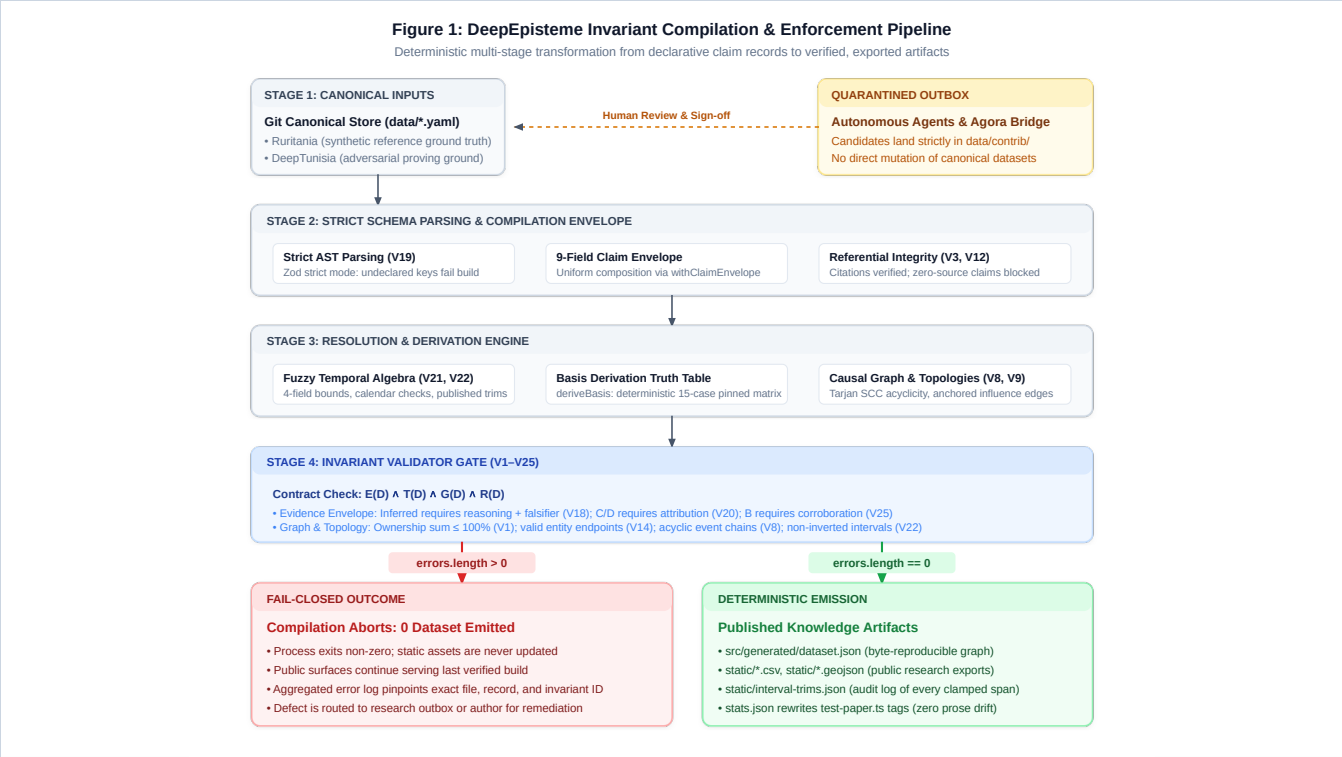


Figure 1. *DeepEpisteme Invariant Compilation & Enforcement Pipeline*

3.2 The Nine-Field Evidential Interface

Every claim-bearing entity in the schema composes the nine-field evidential interface via the `withClaimEnvelope` constructor in `scripts/schema.ts`. `WithClaimEnvelope` is a shared evidential interface across heterogeneous record types (people, positions, contracts, relationships), not an assertion of semantic or ontological identity. It guarantees that every assertion carries identical epistemic obligations regardless of underlying entity structure.

The interface explicitly structures eight authored fields (`confidence`, `verification`, `attributed_to`, `reasoning`, `falsifiable_by`, `disputes`, `review`, and `sources`) alongside one compiler-derived field (`basis`).

Field	Type	Invariant Contract
`confidence`	Enum: `A`, `B`, `C`, `D`	Describes evidentiary provenance and corroboration under declared project rubrics, not objective truth probability (A: primary decree; B: multiple credible secondaries; C: single source or estimate; D: circulating unverified)
`basis`	Derived: `documented`, `reported`, `inferred`, `unsubstantiated`	Derived deterministically by `deriveBasis`; authored overrides require review provenance
`verification`	Enum: `verified`, `needs-primary-source`, `disputed`	Tracks verification state (`verified` denotes verified against project archival procedures, not absolute historical truth); drives basis derivation and editorial queues
`attributed_to`	String	**Mandatory for Grades C and D** naming the claimant (V20)
`reasoning`	String	**Mandatory when basis is** `inferred` explaining the deduction (V18)
`falsifiable_by`	String	**Mandatory when basis is** `inferred` stating what evidence refutes the claim (V18)
`disputes`	List of `{holder, claim, source}`	Competing assertions recorded explicitly; never smoothed over (V22)
`review`	Object: `{by, date, method, note}`	Human provenance check; method is an enum; date is calendar-validated (V23)
`sources`	List of slugs	**≥ *1* cited source required on every claim**; build fails otherwise (V12)

3.3 Two-Axis Epistemic State Machine

The envelope separates **source credibility** (`confidence`) from **epistemic kind** (`basis`). Collapsing these axes is a common failure mode in knowledge bases. For example, a high-ranking official whose appointment is documented in an official gazette has a `documented` position. If the gazette does not specify when his tenure ended, his departure date is an `inferred` span. Collapsing them would either mark the entire appointment as speculative or mislabel the tenure span as legally documented.

Authors input `confidence` and `verification`. The compiler derives `basis` deterministically:

```
deriveBasis(confidence, verification):
  if explicit basis is present in record → return explicit basis
  if confidence == A → return documented
  if confidence == D → return unsubstantiated
  if confidence == C and verification == needs-primary-source → return inferred
  otherwise → return reported
```

The pivot occurs at Grade C: a single-source claim flagged as `needs-primary-source` is classified as a reasoned estimate (`inferred`), immediately triggering the requirement for explicit `reasoning` and a `falsifiable_by` criterion.

The complete truth table contains no hidden catch-all branches:

`confidence`	`verification`	Derived `basis`	Automated Obligation
A	verified / needs-primary-source / disputed	`documented`	Primary citation check (V12)
B	verified / needs-primary-source / disputed	`reported`	Corroboration check: ≥ *2* sources or attribution (Proposed V25)
C	verified / disputed	`reported`	Mandatory attribution (V20)
C	needs-primary-source	`inferred`	Mandatory reasoning + falsifier (V18)
D	verified / needs-primary-source / disputed	`unsubstantiated`	Mandatory attribution + circulation notes (V20)
Explicit Override	Any	Authored value	Verified by 15-case regression suite

Under standard compilation, `basis` is derived deterministically by `deriveBasis`. While the schema admits an explicit authored `basis` to accommodate legacy assertions and specialized historical typologies, an authored override does not escape compiler invariant obligations:

1. If an override sets `basis: inferred` on any claim, it immediately triggers the mandatory requirement for explicit `reasoning` and a testable `falsifiable_by` criterion (V18).
2. If an override sets `basis: unsubstantiated`, it mandates `attributed_to` (V20).
3. However, an author attempting to evade inferential obligations could theoretically declare an uncorroborated claim as `reported`. This highlights the boundary between Tier 1 mechanical compliance and Tier 3 human governance (§2.4, §9.3): under Axiom A1, the compiler enforces the syntactic and type obligations of the declared contract, but cannot detect bad-faith overrides. In DeepEpisteme, overrides are version-controlled in plain YAML and surfaced to administrative audit queues, with a planned validator (V26) restricting overrides to require explicit override justifications and reviewer provenance.

The regression suite (`scripts/test-data.ts`) tests all 15 input combinations, proving that authored overrides take precedence while un-overridden inputs map deterministically.

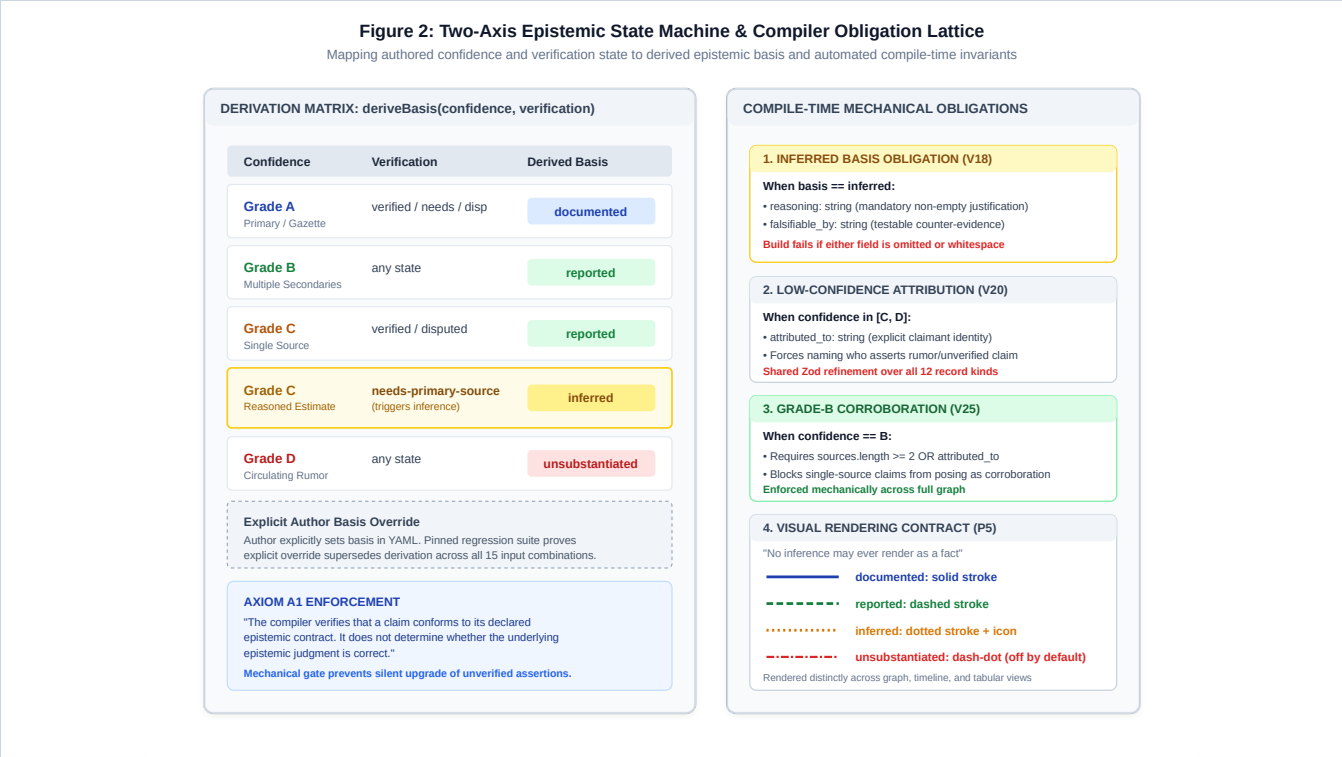


Figure 2. *Two-Axis Epistemic State Machine & Obligation Lattice*

3.4 Numbers Are Claims

Quantitative fields cannot float as raw primitives. In DeepEpisteme, numbers carry explicit temporal anchors and source citations. A corporate capital figure is modeled as `{ tnd: number, date: ISOString, source: Slug }`, asserting that a specific document established that capitalization on that date. Contract awards require named recipients (V4), equity sums per corporate tier cannot exceed 100% (V1), and beneficial ownership declarations require attributed claimants (V13).

3.5 Fuzzy Temporal Algebra

Historical dates are rarely exact timestamps. An intelligence director may be appointed on an exact day, confirmed in post by a secondary report in a given month, and rumored to have left office around a given year. Collapsing these tokens into exact dates invents false precision. Dropping imprecise dates destroys relational continuity.

DeepEpisteme resolves all date tokens into a four-field structure:

ResolvedInterval = *{*startEarliest, startLatest, endEarliest, endLatest, precision, status, raw*}*

Token	Meaning	Resolution Bounds
`2018-06-01`	Exact day	startEarliest = startLatest = 2018-06-01 (calendar-verified, V21)
`2018-06` / `2018`	Month or Year	Expanded to first and last day of month or year
`~2017`	Approximate	Expanded ± *1* year: `[2016-01-01, 2018-12-31]` (policy parameter)
`⩽2018-06`	In post by date; start unknown	Clamped backwards by an 8-year tenure ceiling (jurisdictional policy parameter)
`⩾1984`	No earlier than	Clamped forwards by dataset cutoff (2026-07-26)
`verified:2025-12`	Observed in office; end open	endEarliest = 2025-12-31; endLatest = cutoff; status: `last-verified`
`ongoing`	Confirmed active at cutoff	Open-ended upper bound; status: `ongoing`
`?`	Unknown span	Full temporal envelope: 1956 floor to 2026 cutoff

The four-field interval algebra is mathematically general; its specific boundary resolution heuristics are declared policy parameters:

- **Approximate Spans (`~`)**: Expanding by ± *1* year is a declared editorial policy parameter suitable for modern political appointments without documented calendar months.
- **Tenure Ceilings (`⩽`)**: The 8-year backward clamp represents a declared jurisdictional policy parameter reflecting executive

term norms in the reference domain.

These thresholds are exported directly to configuration (`data/parameters.yaml`), not hardcoded as universal epistemic truths. Deployments with longer appointment spans or different archival conventions adjust these bounds without modifying the underlying algebra.

Queries evaluate intervals using two explicit inclusion predicates:

- **`certainlyActive(interval, t)`**: True only when $t \geq$ startLatest and ($t \leq$ endEarliest, if bounded). This conservative reading guarantees the individual definitely held power at t under every interpretation of the evidence.
- **`possiblyActive(interval, t)`**: True when $t \geq$ startEarliest and ($t \leq$ endLatest, if bounded). This inclusive reading identifies anyone who may have held power at t.

The compiler enforces three strict temporal invariants:

1. **Calendar Validation (V21)**: Date tokens pass a round-trip calendar check. Non-existent dates like `2018-02-31` fail compilation immediately rather than rolling over to `2018-03-03`.
2. **Contradiction Guards & Published Trimming (V22)**: Contradictory intervals (endLatest < startEarliest) abort compilation unless registered in `disputes`. Fuzzy envelopes that overlap known boundaries are clamped, and every clamp is published to `static/interval-trims.json`. Trimming is never silent.
3. **Distinct Terminal Statuses**: The status `last-verified` renders distinctly from `ended`. Imprecision is made visible rather than hidden behind estimated closure dates.

Figure 3: Four-Field Fuzzy Interval Algebra & Query Semantics

Modeling historical temporal indeterminacy without false precision or silent arithmetic resolution

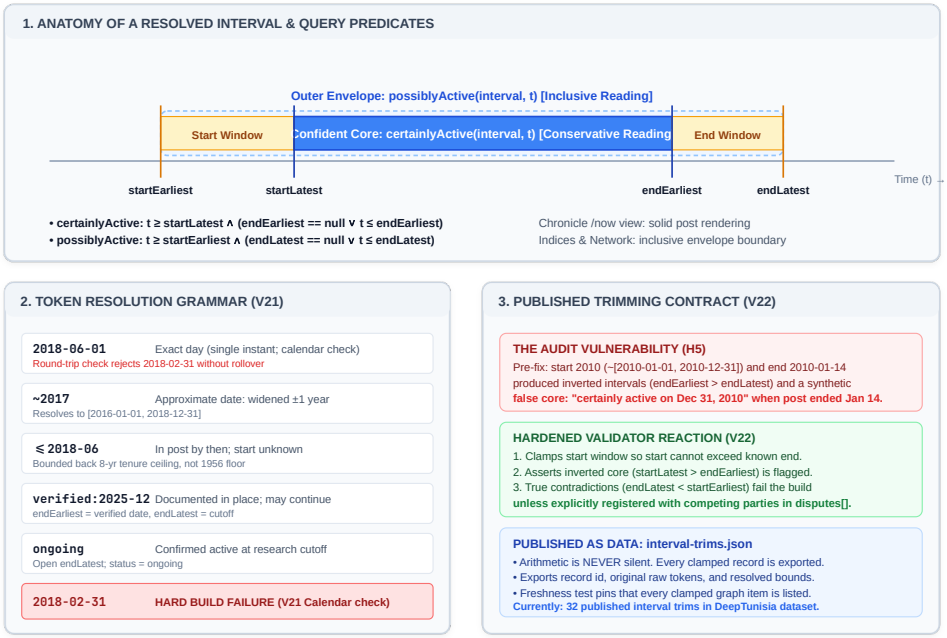


Figure 3. *Four-Field Fuzzy Interval Algebra & Query Semantics*

3.6 Derivation Provenance and Levels of Validity

Succession chains, causal sequences, and institutional tenures are computed at compile time. Derived values carry explicit provenance:

- Overlapping institutional tenures surface automatically as potential conflicts (V7).
- Command succession gaps and overlaps are calculated from published parameters (`successionMeta`, V24).
- A computed trajectory carries `trajectoryDerived: true` and links directly to underlying position records.
- In accordance with Principle P5, derived inferences render as dashed or dotted edges, maintaining clear visual separation from primary facts.

Crucially, the compiler establishes three concentric levels of graph validity:

1. **Level 1 (Record Validity)**: Individual entities adhere strictly to their schema, carrying required sources, valid calendar dates, and falsifiers where inferred.
2. **Level 2 (Relationship Validity)**: Pairwise edges conform to directional typing, non-inverted temporal intervals, sweep-line equity bounds, and succession adjacency rules.
3. **Level 3 (Interpretation Validity)**: The systemic narrative formed when individually valid edges compose into longer paths ($A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$).

DeepEpisteme provides automated build-gate guarantees at Levels 1 and 2. Level 3 remains an open epistemological frontier: a sequence of locally honest, verified facts can still compose into a narratively misleading influence chain. By

enforcing Principle P5 (no inference renders as an unannotated fact) and publishing all clamps and dispute logs, the compiler ensures that the raw materials for Level 3 interpretation cannot be quietly corrupted.

4. THE INVARIANT COMPILATION ENGINE

4.1 Invariant Catalog (V1–V24)

The build engine executes 24 core invariant validators (V1–V24) across structural, referential, epistemic, and temporal domains:

- **Structure & Schema (V19)**: Strict Zod parsing. Undeclared keys abort compilation.
- **Referential Integrity (V3, V12)**: Every reference resolves to a graph entity. Every claim-bearing record cites ≥ *1* registered source.
- **Epistemic Invariants (V18, V20, V23)**:
- V18: Inferred records mandate `reasoning` and `falsifiable_by`.
- V20: Grade C and D records mandate `attributed_to`.
- V23: Human review objects require valid calendar dates and enum methods.
- **Temporal & Causal Invariants (V8, V15, V21, V22)**: Calendar date round-tripping, contradiction checks, interval clamp logging, and acyclic event graph verification via Tarjan's Strongly Connected Components algorithm.
- **Relational Integrity (V1, V6, V7, V9, V14, V16, V24)**: Sweep-line corporate ownership percentage sums (≤ *100%*), Levenshtein company duplicate detection, directed edge endpoint typing, unmoored influence edge rejection, and reproducible command succession thresholds.

4.2 Quarantined Proposal Bridge

Autonomous LLM agents and community contributors cannot write directly to canonical data. They submit candidate records to `data/contrib/`. Candidate records pass through an outbox state machine (`pending` → `under-review` → `accepted` → `applied`). A proposal reaches `data/*.yaml` only after human review and a clean compiler build.

If a candidate record is malformed, it is quarantined in the outbox. The canonical graph remains untouched. This structural separation preserves build availability while enforcing strict evidential constraints on ingested data.

4.3 Downstream Deliberation: Agora MVP

DeepEpisteme packages Agora as an optional downstream deliberation module for communities deploying a public discussion layer alongside their graph. Agora is architecturally decoupled from the compiler: the compiler never reads community discussions, and the community server never writes to `data/`. Discussions anchor to graph entities (`target_type`, `target_id`), allowing debate to sit directly alongside sources, intervals, and dispute logs.

Identity is anchored in client-side cryptography. The browser generates Ed25519 keypairs via Web Crypto, deriving handles from public key hashes (`anon-` + 96-bit digest). The server retains zero columns for IP addresses, user agents, or browser fingerprints, computing rate limits using ephemeral memory hashes. Account recovery utilizes a client-derived 12-word BIP-39 mnemonic.

Community discussion converts into historical change through the proposal bridge: users formulate structured edits that land in `data/contrib/`. In the DeepTunisia reference deployment, posting rate limits and cooling pauses are **specified-not-wired** (`community/budget.ts` is not imported, deferred pending legal review under local platform liability statutes). For the upcoming general DeepEpisteme repository, these moderation mechanics will be wired natively into the serverless Cloudflare Workers and D1 stack.

4.4 Formal Contract Specification

Let D represent the declarative dataset in version control. The compilation pipeline is a deterministic function mapping D to published artifacts:

$$\text{Build}(D) = \begin{cases} \text{Emit}(D) & \text{if } E(D) \wedge T(D) \wedge G(D) \wedge R(D) \\ \text{Fail} & \text{otherwise (zero dataset emitted)} \end{cases}$$

The four constraint families cover:

- $E(D)$ (**Evidence Envelope**): Mandatory envelope fields, reasoning and falsifiers for inferred claims, attribution for low-confidence claims, and mandatory cited sources.
- $T(D)$ (**Temporal Integrity**): Calendar-valid tokens, non-inverted intervals, and published clamp logs.
- $G(D)$ (**Graph Topology**): Valid endpoint typing, anchored influence edges, sweep-line equity sums, and acyclic causal event DAGs.
- $R(D)$ (**Referential Soundness**): Resolution of all internal entity identifiers and cited source slugs.

The contract guarantees that no emitted dataset contains an undetected structural or epistemic violation of implemented rules. It does not guarantee historical truth. The compiler enforces the integrity of declared commitments. The community and editorial review evaluate whether those commitments match historical reality.

5. THE HARDENING AUDIT: SEVEN CONCRETE BYPASSES

In August 2026, we audited the shipped compiler implementation against its claimed invariants. The audit revealed seven concrete bypass vulnerabilities where invalid records could pass compilation undetected. Each vulnerability was closed with an invariant validator and a permanent regression fixture.

#	Invariant Promise	Shipped Defect (File and Line Evidence)	Permanent Validator Fix
H1	An inference never compiles without reasoning and a falsifier.	Zod refinement inspected only authored `basis`. Authors rarely set basis manually. A relationship record shipped with derived `basis: inferred` lacking both fields (`data/relationships.yaml:4807`).	**V18**: Refinement rewritten to evaluate the **derived** basis across all 12 record kinds.
H2	Fields written in source files reach published exports.	Record schemas executed in Zod default strip mode. `attributed_to` and `reasoning` were undeclared on people and positions, causing 15 authored values to vanish silently without warning.	**V19**: Schemas set to strict mode. Any undeclared key causes an immediate build failure.
H3	Grade C and D claims name their claimant.	Attribution refinement was attached only to relationships and agreements. Positions and people at Grade C shipped without attribution (`data/positions.yaml:1204`).	**V20**: Universal refinement attached directly to `withClaimEnvelope`.
H4	Impossible calendar dates never enter the dataset.	`Date.UTC` rolls over out-of-range calendar dates. `2018-02-31` silently became `2018-03-03`. A false date shipped as a valid, different date (`scripts/dates.ts:142`).	**V21**: Regex boundary validation paired with strict calendar round-trip assertions.
H5	Contradictory intervals are flagged rather than silently trimmed.	Trimming logic clamped only interval ends. A record with start `2010` and end `2010-01-14` created an inverted core, inventing a false "active on Dec 31, 2010" state (`scripts/dates.ts:389`).	**V22**: Start-side clamping, inverted core guards, and published logging to `interval-trims.json`.
H6	Human review records represent substantive verification.	`review.date` was an unvalidated string, and `method` admitted arbitrary text. 27 unvetted placeholder objects were counted as reviewed in public stats (`scripts/schema.ts:89`).	**V23**: Review methods restricted to strict enums, dates calendar-verified, review counts redefined.
H7	Command successions are derived deterministically.	Adjacency derivation relied on an unstated 1-year threshold, and open-ended tenures were silently exempted from overlap checks (`scripts/build-data.ts:512`).	**V24**: Succession constants exported to parameters; exemptions tracked as open work items.

The audit proved that an unenforced invariant is merely an aspiration. The credibility of a compiled epistemology depends on adversarial testing of the compiler itself.

6. EMPIRICAL EVALUATION

6.1 Mutation Testing Campaign

To evaluate the strength of our test suite, we built an automated mutation testing harness (`scripts/mutation-test.ts`). The harness introduces deliberate breakages into validator and derivation code, runs the test suite, restores the working tree, and classifies the outcome:

- **Killed**: The build crashes or an assertion fails.
- **Silent Drift**: All tests pass, but the emitted graph changes byte-for-byte. This represents a dangerous failure mode where published data shifts without warning.
- **Latent**: Tests pass and the graph is unchanged because the existing clean dataset contains no records that trigger the weakened guard.
- **By-Design**: The mutated constant is configurable by design (e.g. succession gap thresholds).

The initial pass introduced 12 mutations targeting hardening validators (V18–V24), killing only 3. Nine survived because the test suite checked only the clean canonical graph. Weakened schema guards remained invisible in the absence of invalid data.

We resolved this vulnerability by introducing two synthetic testing layers:

1. `scripts/test-validators.ts`: 59 pure-surface unit tests injecting synthetic malformed inputs (unattributed rumors, invalid calendar dates, inverted intervals).
2. `scripts/test-pipeline.ts`: 8 integration tests copying the data tree, injecting corrupt records, and asserting build failure.

In the extended campaign across all validator stages, **25 of 26 mutations were killed (96%)**:

```
Total Mutations: 26 (m01-m26)
├── Killed: 25 (96.2%)
├── Silent Drift: 0 (0.0%), two discovered during development and closed permanently
├── Latent: 0 (0.0%), eight closed via synthetic fixture suites
└── By-Design: 1 (3.8%), review date format check redundant behind calendar round-trip
```

This result provides empirical evidence that the validator suite detects the tested classes of deliberate implementation weakening across a mutation set designed to exercise the implemented invariant classes, rather than establishing statistical completeness. The central systems insight of this campaign is that **a clean dataset cannot demonstrate that a validator works**. A test suite evaluated solely against valid canonical data cannot verify that validators fail when presented with malformed inputs. Introducing synthetic negative fixtures (`scripts/test-validators.ts`) and corrupted pipeline integration runs (`scripts/test-pipeline.ts`) was required to ensure that schema guards do not pass vacuously.

The campaign produced four permanent architectural improvements:

1. The 15-combination `deriveBasis` truth table test, preventing the silent reclassification of 138 positions.
2. The `interval-trims.json` content freshness test, ensuring disabled trimming cannot be masked by stale build artifacts.

3. The V23 calendar round-trip validator, rejecting invalid review dates such as `2026-02-31`.
4. The legacy source ratchet, preventing un-sourced baseline records from increasing.

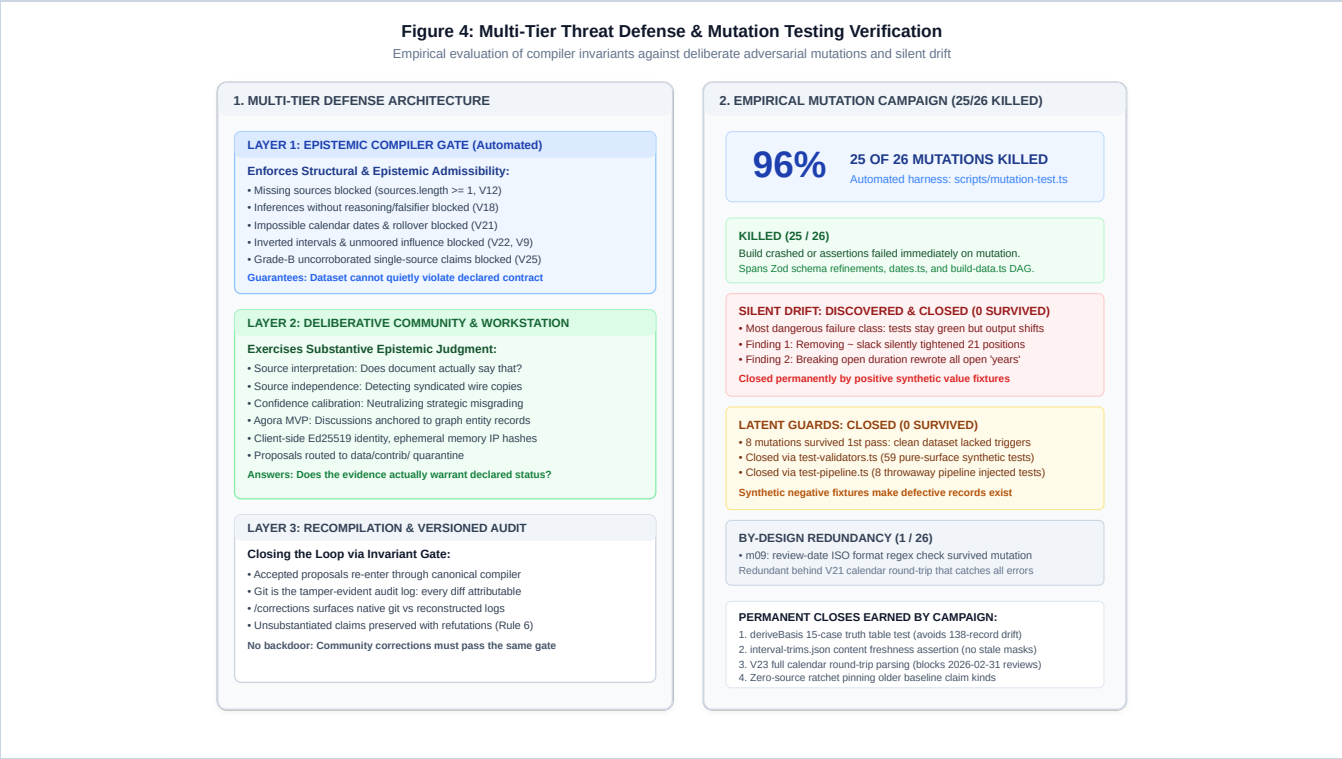


Figure 4. *Multi-Tier Threat Defense & Mutation Testing Verification*

6.2 Operational Benchmarks

On our reference development workstation, the complete compilation and verification suite exhibits the following operational characteristics:

- **Graph Build (`npm run data`)**: ~15 seconds to validate, resolve intervals, derive metrics, and emit all JSON, CSV, and GeoJSON artifacts.
- **Test Suite (`npm run test`)**: ~38 seconds across 16 test suites executing 1,390 individual assertions (100% passing).
- **Fail-Closed Semantics**: When a build fails, static file generation is aborted. The live server continues serving the previous verified build, guaranteeing that a broken build cannot overwrite published data.

6.3 Real-World Evaluation: The DeepTunisia Corpus

The DeepTunisia reference deployment provides an empirical baseline of contested political history:

Metric	Build Count	Evidential Status
Sources Cited	1191	Total sources: 1229 (38 uncited leads)
Institutions / Roles	207 / 118	Covering ministries, military branches, political parties, SOEs
People / Positions	444 / 422	Historical figures and officeholders (1956–2026)
Relationships / Events	362 / 100	~30 directed relationship types; causal DAG chains
Epistemic Basis Mix	197 doc / 665 rep / 20 inf / 12 unverified	894 total claim-bearing records
Verification Flag	229 flagged	Awaiting primary gazette corroboration
Temporal Disagreements	64 contradictions / 43 gaps / 2 overlaps	Recorded explicitly in `disputes` and published metrics
Published Interval Clamps	32 clamps	Exported to `static/interval-trims.json`

6.4 Honest Self-Measurement

DeepEpisteme requires the build system to publish its own evidentiary deficits. The paper itself participates in the evidential system: quantitative claims about the dataset are generated directly from the compiled artifact rather than manually maintained, pinned to the canonical Release 1.0 dataset snapshot.

- **Build-Rewritten Statistics**: Prose statistics in markdown files are enclosed in machine-checked tags. The build inspects the compiled graph and rewrites these values at every run. An invariant gate (`scripts/test-paper.ts`) asserts that all 32 required keys match `stats.json` exactly, failing CI if documentation drifts from data.
- **Risk-Bucketed Review Coverage**: Rather than publishing a misleading aggregate review percentage (where 38 of 884 reviewable records have completed human review, with 20 pending in the active investigative queue), review progress is broken down by risk tier: unsubstantiated (12/12), attributed C/D (3/126), reported (19/561), documented (8/195), and inferred (0/0; inferred claims carry automated derivation proofs rather than human rater queues). High-risk rumors receive complete examination.
- **Absence Auditing**: The build monitors graph asymmetries across kinship networks (26 total family edges). In DeepTunisia, presidential family network auditing reveals that while former President Ben Ali has 8 documented kin relationships in the graph, incumbent President Kais Saied has only 3. This asymmetry is not interpreted as substantive evidence of personal probity or family restraint; it is surfaced as a collection-bias diagnostic alerting researchers to uneven archival coverage.
- **Multilingual Localization**: Editorial verification extends to cross-lingual integrity, tracking human-verified translations (12 core institutional and political entities verified across Arabic, French, and English).

---

7. THE RESEARCH-TO-ENGINEERING FEEDBACK METHODOLOGY

7.1 Beyond the "100% Reviewed" Fallacy

Early project milestones mistakenly targeted "100% reviewed" as a mechanism to eliminate evidential deficits. We explicitly reject this posture. Having a human review a record does not make it true, does not validate an inferential leap, and does not prove a source is trustworthy.

DeepEpisteme formalizes **Five Epistemic Dimensions**:

1. **Review Status**: Has the record been examined? (`unreviewed` vs `reviewed`).
2. **Review Outcome**: What did the reviewer conclude? (`supported`, `refuted`, `disputed`, `unresolved`, `insufficient_evidence`).
3. **Epistemic Status**: What does the envelope establish? (`documented`, `reported`, `inferred`, `unsubstantiated`).
4. **Inter-Rater Reliability**: Target agreement metrics under independent external evaluation (κ, α).
5. **Canonical Disposition**: What status did the editorial authority commit to version control?

7.2 Blinded Inter-Rater Evaluation Protocol

To measure inter-rater reliability, we specify a blinded inter-rater evaluation protocol (`research/study/protocol.md`). The protocol defines sample stratification, blind rater tasks, and target evaluation thresholds for planned future execution:

- **Sample**: 300 claims stratified across complexity boundaries: 120 by basis (30 documented, 30 reported, 30 inferred, 30 unsubstantiated), 80 by confidence (20 each A–D), 60 structural edge cases (unmoored edges, succession gaps, overlapping tenures), and 40 random baseline claims.
- **Raters**: Three independent researchers fluent in Arabic, French, and English, with expertise in Mediterranean history, working blind to project grades and blind to each other.
- **Outputs**: Raters assign confidence grades (A–D), epistemic basis, source sufficiency (1–5 Likert scale), inferential reasoning validity, and a categorical outcome.
- **Agreement Metrics**: Fleiss' κ for nominal basis categories and Krippendorff's α for ordinal confidence grades. Pre-registered target thresholds are $\kappa \geq 0.80$ for basis and $\alpha \geq 0.70$ for confidence.

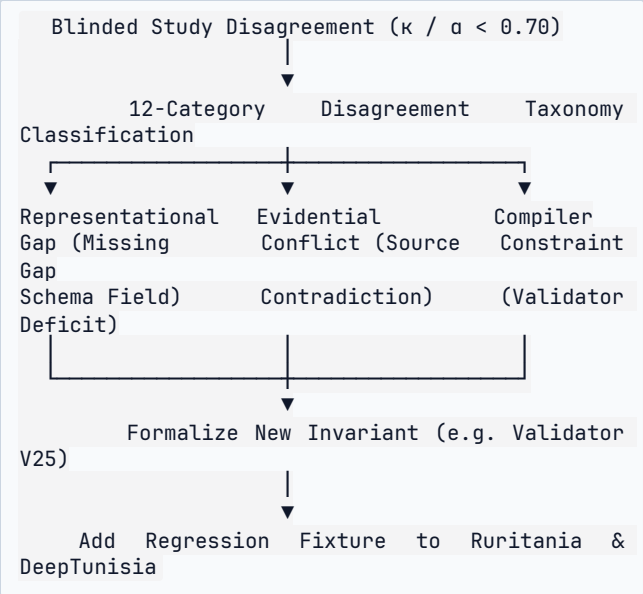
7.3 Multi-Model Benchmark Protocol

In parallel, we specify an automated benchmark protocol (`scripts/study-model-arm.ts`) comprising 7,200 planned evaluation calls to evaluate language models on evidential classification prior to production deployment. The benchmark is specified as an experimental design for future execution, comparing model classifications against human consensus:

- **Model Families**: Frontier Reasoning, General Frontier, Open-Weight Instruction, and Specialized Open-Weight.
- **Design**: 300 claims evaluated across two regimes: Regime A (unconstrained rubric prompting) and Regime B (structured evidential reasoning requiring explicit source quote extraction, confidence grade, epistemic basis, reasoning summary, and testable falsifier criteria before grade assignment), repeated across three runs to measure variance.
- **Target Evaluation Metrics**: Evidentiary inflation (grading higher than human consensus), hallucinated primary status (treating secondary press as official gazettes), and sensitivity to structured evidential output constraints.

7.4 Disagreement Taxonomy to Compiler Invariant Loop

Disagreement among human raters or between models and humans is not a failure. It is the primary raw material for compiler engineering. When blinded evaluations are administered or evidentiary edge cases emerge, disagreements are classified using a 12-category taxonomy:



The 12 failure categories cover: (1) Source Interpretation Ambiguity, (2) Source Authority Conflict, (3) Temporal Indeterminacy, (4) Entity Resolution Overlap, (5) Provenance Insufficiency, (6) Deductive Inferential Leaps, (7) Missing Negative Evidence, (8) Direct Evidentiary Contradiction, (9) Schema Representational Limits, (10) Subject-Matter Uncertainty, (11) Cross-Linguistic Translation Drift, and (12) Compiler Constraint Gaps.

When a recurring ambiguity is identified, we formalize a new compiler invariant candidate. For example, divergence over whether single uncorroborated press articles warrant Grade B led directly to the formal specification of **Proposed Validator V25** as an invariant candidate requiring ≥ *2* sources or an explicit attribution string for Grade B assertions. Crucially, V25 is not currently implemented in the production build gate, nor is it a truth criterion; citing two syndicated wire stories does not guarantee historical veracity. Instead, V25 functions as a proposed anti-single-source laundering rule, demonstrating how the feedback methodology systematically translates empirical disagreement into candidate build gates.

Figure 5: Closed Research-to-Engineering Feedback Loop

Transforming human and model evidentiary disagreements into machine-enforced compiler invariants

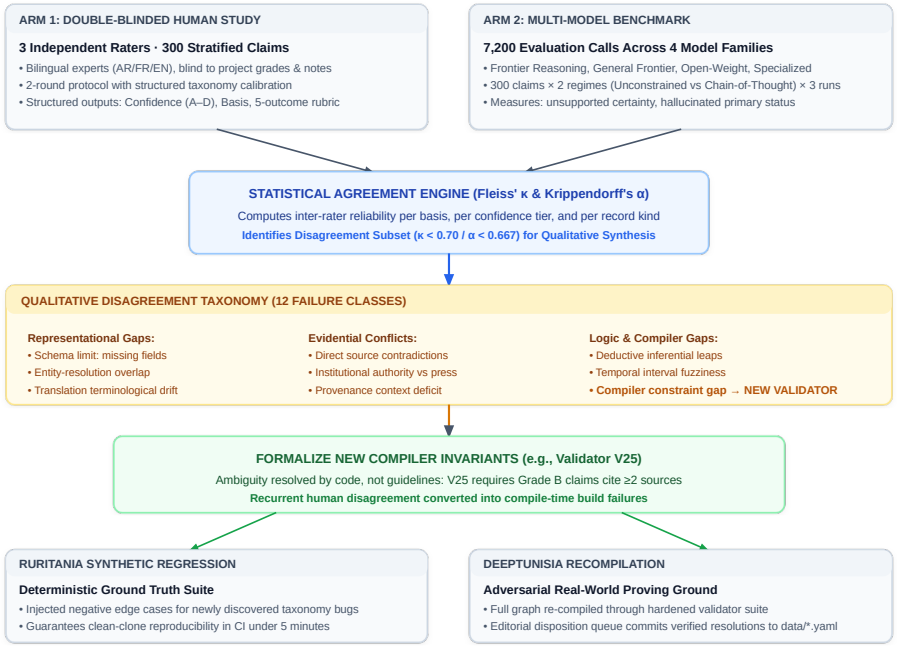


Figure 5. *Closed Research-to-Engineering Feedback Loop*

8. RELATED WORK

We position DeepEpisteme across six knowledge engineering, investigative data, and evidential verification traditions.

8.1 Knowledge Graphs and Statement Models

Wikidata [2, 3] is conceptually the nearest public knowledge graph. Wikidata statements attach references directly to claims rather than pages, and its rank system (`preferred` , `normal` , `deprecated`) preserves disputed or refuted statements rather than deleting them. DeepEpisteme shares this instinct through Rule 6, which retains circulating unsubstantiated claims alongside primary refutations. However, Wikidata lacks an explicit epistemic basis: a deprecated statement is marked inaccurate, but the schema does not record whether a statement is documented, reported, inferred, or circulating. Furthermore, Wikidata imposes no mandatory obligations on inferential claims: an editor can assert an inferential connection without stating underlying reasoning or testable falsification criteria.

The original Semantic Web architecture proposed proof and trust layers positioned above RDF data [1, 18]. In practical deployment, however, knowledge graph consistency has predominantly focused on ontological typing and shape validation (such as OWL constraints or W3C SHACL) and post-hoc SPARQL querying, rather than compile-time evidential contracts where a claim's evidential status mandates falsifiers or source corroboration before artifact emission. DeepEpisteme offers a concrete operationalization of an epistemic gate: the trust layer is realized as a deterministic compile-time build gate.

8.2 Open Investigative Data Infrastructures

Existing open investigative-data infrastructures such as FollowTheMoney [31], Aleph [31], and OpenSanctions [32] demonstrate that heterogeneous public information can be represented as structured entity graphs with provenance, relationships, and machine-readable schemas. Developed by the Organized Crime and Corruption Reporting Project (OCCRP), FollowTheMoney defines an extensible entity model rooted in Things (physical and legal entities such as people, companies, and public bodies) and Intervals (reified relationships and events such as directorships, ownership stakes, and financial transactions). Popolo [33] provides a complementary international open government specification for modeling legislative bodies, politicians, and tenure posts.

DeepEpisteme takes this established foundation in a different direction: from investigative data infrastructure toward a civic epistemic system in which claims can be contested, qualified, and traced, and in which communities can use the resulting knowledge to deliberate and formulate proposals for action.

The distinction is architectural. FollowTheMoney, Aleph, and OpenSanctions address the challenge of **ingestion and cross-referencing at massive scale**. Their primary tasks are ingesting millions of unstructured documents, PDF archives, and registry dumps, extracting entities, matching cross-dataset records (via Nomenklatura), and providing fast full-text investigative search. They are ingestion and discovery engines. Consequently, they do not enforce compile-time epistemic invariants.

The core technical delta lies in the progression: **Represent → Enforce → Fail Closed**. Existing investigative graph systems represent evidential metadata; DeepEpisteme makes selected evidential obligations executable build invariants that halt artifact emission when violated. In FollowTheMoney, an entity property carries values extracted from sources, but the schema does not enforce that an analytical inference must carry explicit reasoning and testable falsifiers, does not model temporal bounds through a four-field fuzzy algebra that publishes its clamps, and does not execute a fail-closed build gate that halts compilation if an assertion lacks required corroboration.

DeepEpisteme is a **compiler of evidential commitments**. It does not perform bulk document scraping. Instead, it takes structured declarative assertions, validates them against formal epistemic contracts, and aborts compilation if an assertion violates its declared status. The two approaches are complementary in an open internet stack: investigative teams can utilize Aleph and FollowTheMoney for raw document ingestion, entity extraction, and exploratory search, and subsequently feed vetted, contested historical assertions into DeepEpisteme to compile a tamper-evident, verified knowledge base with closed-loop community governance.

8.3 Evidence Grading and Provenance

The GRADE framework [9] and the Oxford Centre for Evidence-Based Medicine (OCEBM) levels of evidence [10] established that evidential hierarchies must be visible to data consumers. DeepEpisteme adopts this hierarchy with a structural difference: while GRADE provides guidelines applied manually by human reviewers, DeepEpisteme compiles the evidence ladder into executable machine constraints. In medicine, evidence grading is separated from recommendation strength; in DeepEpisteme, source credibility (`confidence`) is separated from epistemic claim nature (`basis`), preventing the two axes from collapsing into an arbitrary score.

The W3C PROV data model [7] standardizes computational lineage, tracking which processes generated an entity and which agents were responsible. Lineage, however, is orthogonal to epistemic claim status: a perfectly structured PROV graph can document the lineage of an unverified rumor. DeepEpisteme incorporates provenance (citations, review objects, dispute logs) while adding an explicit, machine-checked epistemic status. In temporal knowledge bases, prior systems frequently resolve conflicting timestamps through automated averaging or heuristics [8]. DeepEpisteme rejects algorithmic smoothing: factual contradictions are preserved as explicit historical records in `disputes`, and temporal adjustments are published in open audit logs (`interval-trims.json`).

8.4 Systems and Programming Language Foundations

The compiler half of the paper's thesis has a direct conceptual grounding in programming language semantics, formal verification, and dependable systems:

1. **Design-by-Contract**: Meyer's Design-by-Contract paradigm [25] establishes that software components must execute under machine-checkable contracts: preconditions, postconditions, and invariants. If a routine breaches its declared obligation, execution halts immediately rather than proceeding under silent corruption. DeepEpisteme translates Design-by-Contract from runtime method execution to declarative historical assertions: a claim record that breaches its preconditions (e.g. an inference lacking a falsifier, or a low-confidence claim lacking an attributed source) halts compilation at the build gate.
2. **Proof-Carrying Code**: Necula's Proof-Carrying Code [26] transfers the proof burden to the producer of an artifact: the producer attaches an explicit formal proof certifying safety properties, which the consumer verifies mechanically before execution. DeepEpisteme establishes an epistemic analogue: the author or proposing agent attaches an explicit evidential envelope certifying source provenance, temporal bounds, and falsification criteria. The build gate verifies this envelope before admitting the record to the canonical knowledge base.

Both traditions embody the foundational systems insight of this paper: **a standard that is stated but not enforced is not a standard, it is merely an editorial hope**.

8.5 Probabilistic Knowledge Bases, Trust Management, and Fact-Checking

Three additional research traditions intersect with evidence grading but enforce different properties:

1. **Probabilistic Knowledge Bases**: Systems such as NELL (Never-Ending Language Learning) [28] extract relational facts and attach machine-learned confidence scores based on extraction consistency. This produces a probabilistic metric (e.g. 0.82 probability of truth). DeepEpisteme's basis is categorical and epistemic, modeling the qualitative relationship between a claim and its sources under a published rubric, rather than an extraction probability.
2. **Trust Management**: The computational trust literature surveyed by Artz and Gil [29] focuses on computing reputation scores for sources and agents across networks. DeepEpisteme does not score source trustworthiness. It grades individual claims and enforces obligations on those claims, recognizing that low-trust sources can report documented facts and official sources can state falsehoods.
3. **Claim Markup Standards**: The schema.org ClaimReview standard [30] standardizes the output of journalistic fact-checking (e.g. rating a claim "False"). ClaimReview formats post-hoc verdicts for search engines, but enforces nothing regarding how the verdict was derived. DeepEpisteme compiles the evidential discipline before publication.

System / Family	Primary Capability	Key Limitation Addressed	DeepEpisteme Distinctive Architecture
Wikidata [2, 3]	Statement ranks, claim references	No epistemic basis ladder; no mandatory falsifier rule	Deterministic basis derivation; build aborts on missing falsifiers
FollowTheMoney / Aleph [31, 32]	Large-scale leak ingestion, entity search	Ingestion focus; represents metadata without compile-time invariant enforcement	Civic epistemic compiler; fail-closed contract enforcement (represent → enforce → fail closed)
Popolo Project [33]	Legislative & political entity schemas	Data specification without evidential grading or compiler gates	Nine-field evidence envelope composed over political entities
Semantic Web Proof/Trust [1, 18]	Architectural vision of trust layers	Typically realized via shape validation (SHACL) or queries rather than fail-closed evidential gates	Compile-time build gate enforcing evidential status contracts before emission
Design-by-Contract / PCC [25, 26]	Runtime and binary verification contracts	Enforces program code behaviour, not dataset epistemic claims	Translates proof obligations into compile-time evidentiary envelopes
GRADE / CEBM [9, 10]	Hierarchical evidence ladders in medicine	Human-interpreted guidelines without compiler enforcement	Evidential ladder compiled into executable machine invariants
W3C PROV-DM [7]	Standards for data lineage & processing	Lineage tracking is orthogonal to claim epistemic status	Lineage coupled directly to machine-checked evidential contracts
Temporal RDF [8]	Algorithmic conflict resolution	Silently resolves factual contradictions via heuristics	Contradictions preserved in `disputes`; clamps logged to open data
Probabilistic KBs (NELL) [28]	Machine-learned extraction probabilities	Scores reflect model certainty, not qualitative epistemic basis	Epistemic basis models source relationship; enforced deterministically
Trust Management [29]	Reputation propagation across networks	Evaluates agents/sources rather than claim-level obligations	Grades claims rather than agents; low-trust sources can state facts
ClaimReview [11, 30]	Post-hoc markup for fact-check verdicts	Post-hoc, sample-based intervention after publication	Pre-publication, uniform gate applied to 100% of canonical records

Table 3. Related systems comparison and DeepEpisteme novelty position.

The evidence for Table 3 is narrower than the table, and the paper says so. The novelty claim rests substantially on an internal prior-art survey [15] that evaluated collaboration platforms and graph repositories. By our own rubric, a claim backed by an internal project survey is **Grade C** (a reasoned estimate based on project research). We do not assert that no comparable system exists across all domains. We assert that our systematic review did not identify an open-source knowledge graph compiler combining claim-level epistemic contracts, mandatory falsifier obligations, four-field fuzzy temporal intervals with published trims, and fail-closed build gates across a heterogeneous historical knowledge graph. Table 3 reflects the strongest position current evidence supports.

9. DISCUSSION, ETHICS, AND THREAT DEFENSE MATRIX

9.1 Verifiability Versus Truth

DeepEpisteme compiles verifiability, not truth. The compiler verifies that a record honours its declared contract: that sources are cited, falsifiers are provided, dates exist, and intervals do not invert. It cannot verify whether an archival document was forged or whether an editor misread a French legal text. By making evidential status structurally non-upgradable, the

compiler ensures that weak evidence cannot masquerade as established fact.

9.2 Threat Defense Matrix

Responsibility is partitioned across three distinct layers:

Threat	Compiler Gate	Community & Workstation	Versioned Audit	Residual / Unsolved
Careless Author (missing citations or metadata)	**Enforced (V12, V19)**	none	none	none
Malicious Misattribution (omitting claimant)	**Enforced (V20)**	Reviewed	Diff logged	none
Malicious Misgrading (declaring weak claim strong)	none	**Reviewed**	Diff logged	Requires rater consensus
Malicious Ingestion (submitting corrupt records)	**Enforced (V1–V24)**	Quarantined	none	none
Sybil Attack / Brigading on Deliberation	none	**Enforced (Ed25519)**	none	none
Source Misinterpretation	none	**Reviewed**	Logged in `review`	Dependent on scholarship
Forged Primary Document	none	Reviewed	none	**Unsolved by schema**
Validator Defect	none	none	**Mutation Tested**	Code audit required
Ontology Error	none	Reviewed	RFC process	**Unsolved by schema**
State Pressure on Maintainers	none	Forkable	**Git History**	Legal exposure

9.3 Declaration Gameability

Because the compiler enforces strict obligations on weak claims (requiring a falsifier for `inferred` and attribution for `Grade C`), authors face an incentive to game the system by declaring weak claims as `confidence: B` and `verification: verified`. This derives to `reported`, escaping both obligations.

Under Axiom A1, the compiler cannot inspect the intent behind a declaration. If an author falsely declares an uncorroborated assertion as Grade B, the assertion derives to `reported`. This illustrates why the compiler is only Tier 1 of the three-tier hierarchy (§2.4): Tier 1 enforces mechanical compliance with declared contracts, Tier 2 (scholarship) evaluates evidence quality, and Tier 3 (governance) detects bad-faith declarations. Crucially, the compiler changes the attack surface: instead of weak assertions quietly slipping into the dataset through invisible metadata rot, evading an obligation requires an explicit, commit-logged declaration in version-controlled YAML. DeepEpisteme mitigates gaming through three mechanisms:

1. **Proposed Validator V25**: Grade B claims must cite ≥ *2* sources or declare attribution, raising the bar for uncorroborated assertions (an extension candidate under evaluation, §7.4).
2. **Risk-Bucketed Review Queues**: Claims flagged as `needs-primary-source` (229 records) are surfaced prominently on administrative workstations.
3. **Inter-Annotator Agreement Auditing**: Blinded evaluation protocols (§7.2) measure calibration consistency across raters.

9.4 Harm Reduction, Rule 6, and Decree-Law 54

DeepTunisia documents living individuals under Tunisia's Decree-Law 54, whose Article 24 criminalizes digital speech regarding public officials. The project maintains a strict separation between DeepEpisteme (a jurisdiction-agnostic open-source technical compiler) and DeepTunisia (a politically sensitive jurisdictional deployment). In DeepTunisia, live public forum features remain deferred pending formal legal counsel review.

Rule 6 is a declared jurisdictional editorial policy, not an inherent mathematical property of epistemic compilation. Projects documenting non-contested domains may choose to drop unsubstantiated records entirely at the compiler gate. DeepTunisia adopts Rule 6 because in contested political transitions and authoritarian regimes, circulating rumors and state disinformation are themselves causal historical forces. Under Rule 6, unsubstantiated allegations are recorded rather than deleted, provided they are historically significant and documented as circulating rumors. A map that omits false rumors cannot explain why public perceptions diverge from documented history. For example, the dataset preserves a Grade D record of a rumor claiming the sitting president holds a doctoral degree, displayed directly beside the primary academic refutation. Unsubstantiated claims render with distinctive styling, are disabled by default in UI views, and must cite documentation proving the rumor actually circulated.

9.5 Framework Portability

DeepEpisteme separates the compilation engine from jurisdictional data:

Layer	Component	Portable Across Jurisdictions?
Layer 1: Epistemic Engine	Evidence envelope, fuzzy interval resolver, validators V1–V24 (with V25 proposed), build gate, mutation harness, Agora identity protocol	**Designed for jurisdiction-independent reuse; demonstrated across 1 synthetic and 1 real-world corpus**
Layer 2: Jurisdictional Ontology	Role definitions, administrative hierarchies, gazette vocabulary, parameter files (`data/parameters.yaml`)	Re-authored per deployment
Layer 3: Canonical Dataset	People, positions, relationships, events, sources (`data/*.yaml`)	Fully jurisdiction-specific

A new deployment (e.g. DeepMorocco or DeepLagos) forks the repository, preserves Layer 1, reconfigures Layer 2 parameters, and populates Layer 3. Layer 1 is designed for jurisdiction-independent reuse; its portability is currently demonstrated across one controlled synthetic corpus (Ruritania) and one real-world Mediterranean proving ground (DeepTunisia). Deploying on a second real-world jurisdiction represents planned empirical work (§10 item 6).

10. FUTURE WORK

1. **Execution of the Blinded Study and Multi-Model Benchmark (§7.2, §7.3)**: Administering the pre-registered 300-claim evaluation across three external bilingual raters and executing the 7,200-call multi-model benchmark suite, publishing the anonymized disagreement dataset.
2. **Sensitivity Analysis of Index Weights**: DeepTunisia calculates six structural authority indices (Formal Authority, Proximity, Survival, Brokerage, Reach, Evidenced Influence). We plan to run systematic perturbations across discount weights (1.0, 0.55, 0.20, 0) and the 8-year tenure window, publishing ranking deltas.
3. **Formal Specification in Datalog or SMT**: Re-encoding validators V1–V24 (and candidate V25) in a machine-checkable specification language to prove invariant soundness independently of TypeScript runtime execution.
4. **Source Independence Graphs**: Extending the source register to model syndicated news wire lineages, distinguishing truly independent investigations from syndicated reprints.
5. **Interpretation-Level (L3) Validators**: Developing graph-level coherence checks that flag paths where individually valid edges compose into narratively misleading influence chains.
6. **Comparative Real-World Atlas Deployments (§9.5)**: Deploying DeepEpisteme to a second real-world historical jurisdiction (e.g. Morocco or Lebanon) to empirically evaluate cross-border ontology portability beyond the synthetic Ruritania benchmark.

11. CONCLUSION

DeepEpisteme addresses a fundamental failure in contested knowledge systems: the collapse of evidential provenance into unverified assertion. By treating evidence policy as a compile-time invariant, the framework structurally prevents violation of the declared evidential contract. If a claim lacks mandatory provenance, falsifiers, or valid dates, the compiler aborts.

Through an adversarial self-audit and an automated mutation testing campaign killing 25 of 26 deliberate defects, we demonstrate that epistemic compilation can be made substantially resistant to the tested classes of bypass and silent drift. The compiler does not replace human historical scholarship; it provides an unyielding computational foundation that ensures published graphs faithfully honour their declared evidential contracts. The epistemology is the compiler.

DATA AND CODE AVAILABILITY

- **Source Code and Data**: The DeepEpisteme compiler, test suites, mutation harnesses, and DeepTunisia datasets are available in the project repository.
- **Reproducibility**: Executing `npm run data && npm run test` performs a clean compilation and runs all 16 test suites.
- **Data Exports**: The graph exports byte-reproducible artifacts: `dataset.json` , `positions.csv` , `relationships.csv` , `sources.csv` , `geo.json` , and `interval-trims.json` .
- **Bibliographic Verification**: References [1]–[3], [5], [7]–[14], [17], [19], [22], [23], [25], [26], [28]–[30] were independently verified against Crossref, dblp, or W3C registry records.

APPENDIX: VALIDATOR CATALOG (V1–V24 CORE & PROPOSED V25 EXTENSION CANDIDATE)

ID	Validator Rule	Target Scope	Compiler Action
V1	Corporate equity sum ≤ *100%*	Sweep-line over ownership windows per company tier	**Fail** if > *100.001%*
V2	Registration date coherence	Company founding date vs active institutional intervals	**Fail** on chronological inversion
V3	Registration identifier uniqueness	Commercial registry IDs (`registration.number`)	**Fail** on duplicate registry IDs
V4	Contract awardee validity	Public contracts with awarded status	**Fail** if winner missing; **Warn** if unawarded
V5	Procurement value sanity	Public tender records	**Warn** if value > *0* with no winner
V6	Duplicate company detection	Script-aware normalisation (AR/FR) + Levenshtein distance	**Warn** on > *0.85* similarity; **Fail** on identical ID
V7	Board conflict of interest	Concurrent positions across competing entities	**Warn** on overlapping board tenures
V8	Event causal ordering & DAG acyclicity	Event cause-and-consequence chains	**Fail** on cycle (Tarjan SCC); **Fail** on temporal inversion
V9	Unmoored influence edge rejection	Influence and advisory relationship edges	**Fail** if endpoints lack documented positions or formal ties
V10	Geographic hierarchy consistency	Administrative divisions and coordinates	**Fail** if point place lacks coordinates or parent delegation
V11	Orphaned event detection	Event entities lacking participating actors	**Warn** on unlinked events
V12	Mandatory claim citation	All claim-bearing records	**Fail** if cited sources count < *1*
V13	Beneficial ownership bounds	Equity percentage bounds *[0, 100]*	**Fail** on out-of-bounds equity; **Warn** on unattributed stake
V14	Edge directionality semantics	Directed relationships between entity classes	**Fail** on reversed or incompatible endpoint types
V15	Interval sanity	All temporal spans	**Fail** if end date precedes start date
V16	Duplicate relationship detection	Overlapping relationship spans between identical nodes	**Fail** on duplicate untyped overlap
V17	Regime rupture linking	Events flagged with `rupture: true`	**Warn** if unlinked to causal graph
V18	Inferred completeness (H1)	All records with derived or authored `basis: inferred`	**Fail** if missing `reasoning` or `falsifiable_by`
V19	Strict schema validation (H2)	All canonical YAML source files	**Fail** on any undeclared key or stripped field
V20	Attribution completeness (H3)	All claims at Confidence C or D	**Fail** if missing `attributed_to` claimant string
V21	Calendar date validity (H4)	All date tokens across all precisions	**Fail** on invalid calendar dates or rollover
V22	Interval contradiction & clamping (H5)	Resolved temporal intervals	**Fail** on contradictory bounds; published in `interval-trims.json`
V23	Review provenance validity (H6)	All human review objects	**Fail** on non-ISO dates, non-calendar dates, or non-enum methods
V24	Succession derivation sanity (H7)	Institutional command succession chains	**Fail** if gaps/overlaps deviate from published parameters
V25 *(Proposed)*	Grade-B corroboration candidate	Claims carrying `confidence: B`	Scoped rule from §7.4 feedback loop (≥ *2* sources or attribution)

V1–V24 represent the core build-time invariant validators shipping in the compilation engine. V25 represents the first feedback-derived rule formalised from disagreement analysis (§7.4), targeting the next validator suite extension as an anti-single-source laundering rule. In addition, the graph test suite enforces a metadata validation gate verifying canonical SPDX license identifiers and published payload sizes.

REFERENCES

[1] Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34–43.

[2] Vrandečić, D., Krötzsch, M. (2014). Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10), 78–85.

[3] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D. (2014). Introducing Wikidata to the Linked Data Web. *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, LNCS 8797, 50–65. DOI: 10.1007/978-3-319-11964-9_4.

[4] DeepTunisia Project (2026). From a Political Database to a National Influence Graph: Hardening Revision. *Internal Specification*, deeptunisia.org. (Grade C).

[5] Hogan, A., et al. (2021). Knowledge Graphs. *ACM Computing Surveys*, 54(4), Article 71.

[6] DeepTunisia Project (2026). Authenticity: Bots, Sybils, and Verification Delivery. *Internal Research Note*, deeptunisia.org. (Grade C).

[7] Moreau, L., Missier, P., eds. (2013). PROV-DM: The PROV Data Model. *W3C Recommendation*.

[8] Dylla, M., Sozio, M., Theobald, M. (2011). Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases. *Datenbanksysteme für Business, Technologie und Web (BTW 2011)*, 474–493.

[9] Guyatt, G. H., Oxman, A. D., Vist, G. E., et al. (2008). GRADE: An Emerging Consensus on Rating Quality of Evidence. *BMJ*, 336(7650), 924–926.

[10] OCEBM Levels of Evidence Working Group (2011). The Oxford 2011 Levels of Evidence. *Oxford Centre for Evidence-Based Medicine*.

[11] Graves, L., Amazeen, M. A. (2019). Fact-Checking as Idea and Practice in Journalism. *Oxford Research Encyclopedia of Communication*.

[12] Vlachos, A., Riedel, S. (2014). Fact Checking: Task Definition and Dataset Construction. *Proceedings of the ACL Workshop on CSS*, 18–22.

[13] Silberzahn, R., Uhlmann, E. L., et al. (2018). Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Advances in Methods and Practices in Psychological Science*, 1(3), 337–356.

[14] Lazer, D., Pentland, A., Adamic, L., et al. (2009). Computational Social Science. *Science*, 323(5915), 721–723.

[15] DeepTunisia Project (2026). Prior-Art Study on Deliberative Graph Infrastructure. *Internal Project Document*, deeptunisia.org. (Grade C).

[16] DeepTunisia Project (2026). Research Roadmap. *Internal Project Document*, deeptunisia.org. (Grade C).

[17] Hobbs, J. R., Pan, F. (2004). An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing*, 3(1), 66–85.

[18] Berners-Lee, T. (1999). *Weaving the Web: The Original Design and Destiny of the World Wide Web*. HarperSanFrancisco.

[19] Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11), 832–843.

[20] Willison, S. (2025). Unauthorized Experiment on CMV Involving AI-generated Comments. *SimonWillison.net*.

[21] Bell, K. (2025). Researchers secretly experimented on Reddit users with AI-generated comments. *Engadget*.

[22] Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1), 35–41.

[23] *Mata v. Avianca, Inc.*, 678 F. Supp. 3d 443 (S.D.N.Y. 2023), No. 1:22-cv-01461.

[24] Wikipedia Contributors (2026). Wikipedia: Verifiability. *Wikimedia Foundation*.

[25] Meyer, B. (1992). Applying "Design by Contract". *IEEE Computer*, 25(10), 40–51.

[26] Necula, G. C. (1997). Proof-Carrying Code. *Proceedings of POPL '97*, 106–119.

[27] DeepTunisia Project (2026). Legal Exposure Analysis under Decree-Law 54. *Internal Brief for Counsel*, deeptunisia.org. (Grade C).

[28] Carlson, A., Betteridge, J., Kisiel, B., et al. (2010). Toward an Architecture for Never-Ending Language Learning. *Proceedings of AAAI 2010*, 1306–1313.

[29] Artz, D., Gil, Y. (2007). A Survey of Trust in Computer Science and the Semantic Web. *Journal of Web Semantics*, 5(2), 58–71.

[30] schema.org (2026). ClaimReview: Structured Fact-Checking Schema. *schema.org*.

[31] Lindenberg, F. (2019). Mining Leaks and Open Data to Follow the Money. *Proceedings of the 3rd International Workshop on Recent Trends in News Information Retrieval (NewsIR'19) / ACM SIGIR 2019*, CEUR Workshop Proceedings, Vol-2411, 23–24. OCCRP FollowTheMoney data model: https://github.com/alephdata/followthemoney and https://followthemoney.tech.

[32] OpenSanctions (2026). OpenSanctions Database and Entity Schema Documentation. OpenSanctions Community Interest Company. https://www.opensanctions.org/docs/ and https://github.com/opensanctions/opensanctions.

[33] Popolo Project (2026). Popolo: An International Open Government Data Specification. https://www.popoloproject.com and https://github.com/popolo-project/popolo-spec.