# AEGIS PRIME

**Structural verification of judicial dispositions — deterministic set-membership checks that catch AI errors text-containment cannot.**

---

**Author:** Richard L. Sanders · Utah Bar #15728 **Role:** Sole inventor and implementer **Version:** 1.0 — 2026-05-13 **Companion paper:** `docs/whitepapers/01-deterministic-closed-corpus-verification.md` (AEGIS, the text-containment layer) **Source:** `lib/case-fact-extractor.js`, `lib/verification-pipeline.js`, `lib/post-stream-gate.js`, `routes/admin/aegis-prime-verify.js`, migration `179`, in the public repository at <https://github.com/Benchslap/Benchslap>

---

## Abstract

AEGIS (paper 01) is the first half of BenchSlap's verification architecture: every authority row carries a SHA-256 content hash and a 5-gram shingle signature, and every AI-generated citation is checked for containment in the canonical opinion text. That defeats the "fabricated quote" class of hallucination — the AI cannot put words in a court's mouth that the court did not say.

AEGIS PRIME is the second half. It addresses the failure mode AEGIS cannot catch: **the AI cites a real case, quotes its real text, and gets the structural conclusion exactly wrong.** It says the case was "affirmed" when the case was reversed. It quotes the dissent and frames it as the holding. It cites a case that was overruled three years ago as if it were still good law. The text-containment check passes — the quote really is in the opinion. The legal claim is still false.

AEGIS PRIME pre-extracts structured facts about every Utah case at ingest time (disposition, panel vote, holding binding weight, treatment graph). At verification time, AI claims about those facts are checked against the fact store, not against the opinion text. The check is deterministic set-membership; the trigger is a hard block when the AI's structural claim does not match the stored fact.

This paper describes the schema, the regex-first / LLM-platinum two-layer extraction pipeline, the four hard-block matrix entries, and the architectural reason these checks are not feasible to retrofit onto a conventional generate-then-verify legal AI.

---

## 1 · The problem AEGIS does not catch

A motion drafter generates this paragraph:

> *Counsel relies on State v. Smith, 2021 UT 12, in which the Utah Supreme Court **affirmed** the conviction over a substantial-evidence challenge.*

AEGIS runs. The citation `2021 UT 12` resolves to a real opinion. The quoted phrase "affirmed the conviction" appears verbatim in the text of the opinion — which is the dissent. The majority reversed. AEGIS gives the citation a GREEN flag because text-containment passes.

The user files the motion. Opposing counsel responds: "Counsel cites *Smith* as affirming the conviction. *Smith* reversed the conviction. Counsel either misread the opinion or is attempting to mislead this Court." The judge agrees. The drafter is sanctioned under URCP 11(b)(2).

Text containment is the wrong check. The right check is: **is *Smith*'s disposition AFFIRMED or REVERSED, and does the drafter's claim agree?**

That is a structural check. AEGIS PRIME runs it.

---

## 2 · The schema

Migration `179` extends `authority_analysis` with structured-fact columns. Every Utah opinion is pre-extracted to populate them at ingest time. The columns:

| Column | Type | Meaning |
|---|---|---|
| `disposition` | enum | `AFFIRMED`, `REVERSED`, `VACATED`, `REMANDED`, `AFFIRMED_IN_PART_REVERSED_IN_PART`, `DISMISSED`, `MODIFIED`, `OTHER`, `NOT_EXTRACTABLE` |
| `panel_vote` | jsonb | `{ majority: [judge names], concurrence: [...], dissent: [...] }` |
| `holding_binding_weight` | enum | `BINDING`, `PERSUASIVE`, `DICTA`, `REJECTED`, `SUPERSEDED` |
| `superseded_by_opinion_id` | int FK | If the case has been overruled, points at the overruling opinion |
| `case_treatment_graph` | jsonb | Directed edges: `[{type: 'OVERRULED' \| 'SUPERSEDED' \| 'DISTINGUISHED' \| 'NEGATT target_o source_c M, sourc '...' }]` |

The treatment graph is bidirectional. If opinion A overrules opinion B, then A has an outgoing edge `OVERRULED → B` AND B has an incoming edge `OVERRULED_BY ← A`. The graph is populated by a separate batch process (`scripts/build-treatment-graph.js`) that scans every opinion's text for explicit overrule / supersede / distinguish phrasing and adds the corresponding edges.

---

## 3 · Two-layer extraction

AEGIS PRIME extracts the structured facts in two passes:

**Layer 1 — Regex** (`lib/case-fact-extractor.js`). Pattern-matches on the standard Utah-opinion phrasing: `"We affirm"`, `"The decision of the trial court is reversed"`, `"We vacate and remand"`, `"Judgment affirmed in part and reversed in part"`, etc. Zero LLM cost. Roughly 80% extraction coverage at the disposition level; near-100% for the panel-vote signature (judges' names appear in a structured header that regex parses reliably).

```
const DISPOSITION_PATTERNS = [
    { pattern: /\bwe affirm\b/i,              disposition: 'AFFIRMED' },
    { pattern: /\bwe reverse\b/i,             disposition: 'REVERSED' },
    { pattern: /\bwe vacate\b/i,              disposition: 'VACATED' },
    { pattern: /\bwe remand\b/i,              disposition: 'REMANDED' },
    { pattern: /affirmed in part.{0,30}reversed in part/i, disposition:
'AFFIRMED_IN_PART_REVERSED_IN_PART' },
    // ... 14 more patterns
];
```

The Layer 1 extractor is deliberately strict: when the patterns do not unambiguously match, it returns `NOT_EXTRACTABLE` rather than guessing. Bad guesses corrupt the fact store; missed extractions are caught by Layer 2.

**Layer 2 — Platinum LLM** (`scripts/extract-platinum.js`). For opinions Layer 1 flagged `NOT_EXTRACTABLE`, plus a periodic re-scan of high-importance opinions for richer extraction (holdings, summaries, topic taxonomy, notable quotes, key authorities cited), the Platinum extractor runs the opinion text through Gemini Flash with a structured-output schema:

```
{
    "disposition": "AFFIRMED|REVERSED|...",
    "holding": "string — the one-sentence rule the court announced",
    "summary": "string — three-sentence procedural posture + outcome",
    "topics": ["string"],
    "notable_quotes": [{ "text": "string", "pinpoint": "string" }],
    "key_authorities_cited": [{ "citation": "string", "treatment": "string" }],
    "panel_vote": { "majority": [...], "concurrence": [...], "dissent": [...] }
}
```

The LLM output is itself validated against the regex-extracted ground truth before it is written to `authority_analysis`. If Layer 1 returned `AFFIRMED` and Layer 2 returns `REVERSED`, the row is flagged for manual review rather than silently overwritten. The two layers cross-check each other; the human is the tiebreaker.

---

## 4 · The hard-block matrix

`lib/post-stream-gate.js` is where AI output meets AEGIS PRIME. After every AI stream completes and before the response is shown to the user, the gate scans for citation-bearing claims and asks the verification pipeline to check each one against the structured facts. The check returns one of these verdicts:

| Verdict | Trigger | Action |
|---|---|---|
| `DISPOSITION_MISMATCH` | AI says "the court affirmed" but `disposition = REVERSED` (or vice versa) | HARD BLOCK — response rewritten with the correct disposition; warning surfaced |
| `ATTRIBUTION_MISMATCH` | AI attributes a quote to the majority but the quote appears only in the dissent | HARD BLOCK — claim flagged; user shown which opinion the quote actually comes from |
| `BINDING_WEIGHT_MISMATCH` | AI cites dicta as binding holding | HARD BLOCK — quote re-tagged as dicta; user warned that the rule the AI announced is not the case's holding |
| `SUPERSEDED_CASE` | AI cites a case where `superseded_by_opinion_id` is non-null | HARD BLOCK — citation rewritten with the overruling case; warning that the cited authority is no longer good law |

The block is genuine: the user does not see the original AI response. They see the corrected version with a banner explaining the substitution. If the AI is uncertain about a disposition — because the case is from a jurisdiction we have not fully extracted — the gate degrades to a soft warning (`HOLDING_UNVERIFIED`) rather than a block. False positives on extracted Utah cases are very rare in practice because the schema is conservative: when Layer 1 cannot determine the disposition unambiguously, the row stays `NOT_EXTRACTABLE` and the verification falls through to AEGIS text-containment with no structural check.

Pass silently (no block, no warning): `VERIFIED`, `NO_CLAIMS_DETECTED`, `null` (facts pending extraction for this jurisdiction).

---

## 5 · Production state and operational metrics

Deployed 2026-04-10 with migration `179`. Extraction state as of 2026-05-13:

- **33,673 dispositions** populated across the corpus (Layer 1 + Layer 2 combined; verified by live `SELECT COUNT(*) FROM authority_analysis WHERE disposition IS NOT NULL`)
- **115 overruled cases** with non-null `superseded_by_opinion_id`

- Treatment graph edges continue to be populated by the batch overrule scanner — initial Utah pass landed 238 edges; corpus-wide pass is incremental

The admin dashboard at `/aegis-prime-demo.html` exposes the verify endpoint directly:

```
POST /api/admin/aegis-prime/verify
{
    "citation": "2021 UT 12",
    "claim": "the court affirmed the conviction"
}

Response:
{
    "verdict": "DISPOSITION_MISMATCH",
    "stored_disposition": "REVERSED",
    "ai_claimed_disposition": "AFFIRMED",
    "action": "HARD_BLOCK",
    "substitute_text": "the court reversed the conviction"
}
```

`GET /api/admin/aegis-prime/stats` returns rolling per-tool counts of disposition mismatches caught, attribution mismatches caught, and superseded-case blocks. The numbers are telemetry; they do not feed back into the gate's decision.

---

## 6 · Why this is not a generate-then-verify retrofit

A conventional legal AI pipeline answers the question "did the model hallucinate?" with a second LLM call that asks "is this output consistent with the cited source?" That approach has three architectural flaws AEGIS PRIME is built to fix:

**Flaw 1 — Same-vendor circular check.** Asking GPT to verify GPT's output, or asking Claude to verify Claude, is asking the same statistical surface to second-guess itself. When the model is confidently wrong, the verifier is confidently wrong in the same direction. AEGIS PRIME's check is deterministic set membership: the disposition column either equals `AFFIRMED` or it doesn't. No LLM is consulted at verify time.

**Flaw 2 — Source-text re-summarization.** "Read this opinion and tell me if the AI summarized it correctly" sends the opinion text back through an LLM. The LLM can make a different mistake on the same text. AEGIS PRIME never re-asks the LLM. The structured facts were extracted ONCE, were validated by the two-layer cross-check, and are now load-bearing data — not regenerated on every query.

**Flaw 3 — Verifier latency.** A second LLM call adds 2-5 seconds per citation and can fail or rate-limit. AEGIS PRIME's verify path is a single PostgreSQL SELECT keyed on `(neutral_citation)` or `(reporter_citation)` — sub-millisecond at production scale. The verification cost is a constant added to the response time, not a multiplier.

The architectural commitment that enables AEGIS PRIME is the same one that enabled AEGIS: **the corpus is closed, the facts are pre-extracted, and the verify path is a database lookup, not a model call.** A vendor whose architecture is "generate, then verify with a second model" cannot retrofit AEGIS PRIME without first ingesting the corpus, then writing the extraction pipeline, then re-writing the gate to consult the fact store. By the time those three changes are made, they have rebuilt AEGIS PRIME.

---

## 7 · Novelty claims

1. **Structured-fact schema for judicial dispositions, panel votes, holding binding weights, and treatment graphs** populated at corpus-ingest time, queryable as deterministic set-membership at AI-output-verification time.
2. **Two-layer extraction pipeline** in which a regex-first pass (`case-fact-extractor.js`) handles the high-coverage cases and a Platinum LLM pass (`extract-platinum.js`) handles the long tail,

with cross-validation between layers preventing silent overwrites.

3. **Hard-block matrix** (`post-stream-gate.js`) in which four named verdicts — `DISPOSITION_MISMATCH`, `ATTRIBUTION_MISMATCH`, `BINDING_WEIGHT_MISMATCH`, `SUPERSEDED_CASE` — rewrite AI output before display rather than warn after the user has read it.
4. **Treatment graph as authoritative overrule index**, populated by batch text scanning rather than license to a commercial citator service, allowing the closed-corpus pipeline to determine whether a cited case is still good law without external dependency.

The four elements in conjunction are AEGIS PRIME. The companion paper (01) covers the text-containment layer (AEGIS); together the two layers form the verification gate that distinguishes BenchSlap from generate-then-verify products.

---