The V4 Advocacy Authority Algorithm

A deterministic-first, neuro-symbolic truth-table engine for legal argument verification.

Author: Richard L. Sanders · Utah Bar #15728 **Role:** Sole inventor and implementer
Version: 1.0 — 2026-05-13 (engine v5.6.4) **Source:** `lib/advocacy-algorithm.js` (3,471 lines), with subordinate modules `lib/deterministic-logic.js`, `lib/peer-review.js`, `lib/dempster-shafer.js`, `lib/bayesian-engine.js`, `lib/argumentation-framework.js`, `lib/defeasibility.js`, `lib/burden-of-proof.js`, `lib/toulmin-parser.js`, `lib/enthymeme-detector.js`, `lib/gate-a-presort.js`, `lib/multi-model-v4.js`, in the public repository at <https://github.com/Benchslap/Benchslap>

Abstract

Every commercial AI-assisted legal-research tool today is, at its core, a single large language model with prompt scaffolding. The model produces an argument, the wrapper presents it to the user, and the user is told to trust the output. There is no algorithmic check on whether the argument is actually valid — whether its law is real, its facts are in the record, its logic follows from its premises, or its conclusion is supported by all three.

The V4 Algorithm — the engine I call **A4** internally and **V4** externally — is a deterministic-first verification layer that sits on top of the model. It treats every legal argument as a three-bit truth table over (law present? fact present? logic valid?) and dispositively classifies the argument into one of nine cells, each with a defined color flag, a defined remedy, and a defined safe-to-cite verdict. The model is allowed to produce drafts; V4 decides whether those drafts may ship.

This paper describes the V4 truth table, the four-gate decomposition pipeline, the eight subordinate calculi V4 runs in parallel against every argument, and the deterministic-first design rule that distinguishes V4 from every other AI-assisted legal-research engine. The rule is simple and uncompromising: **deterministic logic runs first; the Bayesian and neural layers may never override its verdict.**

---

1 · The problem

A legal argument has three irreducible components:

1. **Law** — a citation to authority that, as a matter of black-letter law, supports the rule the argument advances.
2. **Fact** — a factual assertion that, as a matter of evidentiary record, places the argument's subject within the rule's scope.
3. **Logic** — the inference connecting law to fact, drawing a conclusion that the rule applies to this fact.

Every commercial AI legal tool emits all three at once, in prose, with no machine-readable separation. The reader cannot tell whether the model invented the citation, fabricated the fact, or chained a non-sequitur from real law to real fact. The reader has to read carefully, check every citation manually, and trust the model on the logic. Most readers cannot. Most readers do not.

I refused to ship a tool that asked the user to trust the model on the logic. V4 is the result.

---

2 · The truth table

V4 decomposes every argument into the three bits `P(Law) × Q(Fact) × R(Logic)`. Each bit takes one of three values: 1 (verified present), 0 (verified absent), or 0.5 (partially present / unverified). The cross-product is nine cells, each with a dispositive name, color flag, and verdict (from `lib/advocacy-algorithm.js`):

P · Q · R	Status	Flag	Meaning
1 · 1 · 1	VERIFIED	**GREEN**	Sound argument. Law, fact, and logic all confirmed. Safe to cite.
1 · 1 · 0.5	PERMISSIBLE	BLUE	Arguable inference. Facts support the conclusion but do not require it. Safe to argue, with caveat.
1 · 1 · 0	NON_SEQUITUR	RED	Logic error. Law and fact are confirmed but the inference does not follow. **Flag for review.**
1 · 0 · 1	FABRICATION	BLACK	Fact not in record. The argument cites real law but the fact it depends on is not in the evidentiary record. **Critical — do not cite.**
1 · 0 · 0	FABRICATION	BLACK	Fact + logic failure. Critical.
0 · 1 · 1	FICTION	BLACK	Invalid law. Fact and logic are sound but the cited authority is hallucinated, overruled, or misstated. Critical.
0 · 1 · 0	FICTION	BLACK	Law + logic failure. Critical.
0 · 0 · 1	CRITICAL_FAIL	BLACK	Only logic is valid; nothing else.
0 · 0 · 0	CRITICAL_FAIL	BLACK	Total failure.
1 · 0.5 · 1	UNVERIFIED	YELLOW	Law valid, logic valid, fact unconfirmed. Hold for confirmation.
0.5 · 0.5 · 0.5	UNVERIFIED	YELLOW	Any combination of unconfirmed states. Hold.

GREEN ships. BLUE ships with caveat. YELLOW holds. RED reviews. BLACK refuses.

The classification is **per-argument**, not per-document — a 10-page motion may have 47 arguments, each of which receives its own three-bit classification and color flag. The user sees a document-level pass/fail summary plus a per-argument table; the user can drill into any RED or BLACK cell to see exactly which bit failed and why.

---

3 · The four gates

V4 runs every input through four sequential gates. Each gate has a specific, narrow job, and the next gate only sees inputs that have already passed the previous one. A failure in any gate is a definitive failure — it does not bubble up to the next gate as "maybe."

Gate A · Presort

`lib/gate-a-presort.js` classifies the input into one of seven argument-type buckets before decomposition: rule-application, statutory-interpretation, constitutional-claim, balancing-test, factual-determination, procedural-motion, or undefined. The classification picks one of seven specialized decomposition prompts (`lib/gate-a-prompts.js`) that follows the analytical structure courts actually expect for that argument type. A constitutional claim does not get the same decomposition treatment as a procedural motion; the prompt for each is hand-tuned to the analytical structure of that genre.

Token-savings estimator (`estimateTokenSavings()`) shows the specialized prompts use 35-60% fewer tokens than a single all-purpose prompt would, because they ask only the questions relevant to the specific argument type.

Gate B · Decomposition

Gate B extracts the three bits. For each argument it identifies:

- The cited authority (citation extraction → `lib/verification-pipeline.js` → AEGIS hash check).
- The cited facts (fact extraction → record-position pin).
- The inference rule connecting the two (deductive form, analogical mapping, policy argument, or analogical-from-distinguishing).

This is the only gate where an LLM produces structured output. Every other gate operates on the structured output, deterministically.

Gate C · Verification

Gate C operates on the structured output. For each bit:

- **P(Law)** — citation verified through the five-tier pipeline (T0 local cache, T1 Utah Courts .gov, T2 CourtListener, T3 Caselaw Access Project, T4 multi-model peer review) plus AEGIS content-hash check (the cited holding's text must be substring-or-shingle-contained in the corpus's stored opinion text). Hard-block on AEGIS tamper or AEGIS PRIME disposition mismatch.
- **Q(Fact)** — record citation verified against the silo's document index via the fact-freezer. The fact must be locatable at the cited record pin; if it is not, Q = 0.

- **R(Logic)** — inference form validated against the discrete-math rules of inference. Modus ponens, modus tollens, hypothetical syllogism, disjunctive syllogism, and analogical mapping each have explicit form requirements. Fallacies are flagged by name from `data/fallacy_library.json`. R = 0 returns the specific fallacy as a citation in the report.

Gate C is **deterministic**. It does not call any LLM. It does not infer anything. It runs the rules of inference against the structured output Gate B produced, and it outputs a hard bit.

Gate D · Synthesis

Gate D combines the three bits into the truth-table classification, emits the color flag, and writes the verdict to the user. It also runs the eight parallel calculi (below) for cases where the bit pattern is ambiguous (anything with a 0.5).

---

4 · The eight parallel calculi

For every argument that arrives at Gate D with at least one partial bit (P = 0.5, Q = 0.5, or R = 0.5), V4 runs eight subordinate calculations in parallel and uses their consensus to refine the partial bit:

1. **Deterministic logic** (`lib/deterministic-logic.js`) — formal-logic rules of inference. The first calculus to fire, the last to lose. Per Richard's standing order: deterministic logic never overrides to GREEN, only to a more conservative cell. This is the **deterministic-first design rule**: when the deterministic engine says "invalid," no probabilistic engine downstream may say "valid."

1. **Dempster-Shafer belief functions** (`lib/dempster-shafer.js`) — belief assignment over the possibility space when multiple sources of evidence partially support overlapping conclusions. Resolves partial-bit conflicts where two sources both support but neither alone is sufficient.

1. **Bayesian epistemological layer** (`lib/bayesian-engine.js`) — probabilistic reasoning between Dempster-Shafer and the final verdict. Allows the system to express "given prior P(holding true) and evidence E, posterior is X" as a number, never as a category.

1. **Defeasibility classifier** (`lib/defeasibility.js`) — five-level defeasibility ranking (binding-rule / rebuttable-presumption / default-rule / general-policy / weak-heuristic). Higher defeasibility means lower confidence the argument survives counter-argument. Flagged in the report.

1. **Burden-of-proof evaluator** (`lib/burden-of-proof.js`) — selects the applicable burden (preponderance / clear-and-convincing / beyond-reasonable-doubt /

substantial-evidence / arbitrary-and-capricious) and asks whether the argument's evidentiary support meets it. This is the only calculus that ties the bit pattern back to the procedural context (civil / criminal / administrative).

1. **Toulmin warrant parser** (`lib/toulmin-parser.js`) — extracts the Toulmin diagram (claim / data / warrant / backing / qualifier / rebuttal) and validates the warrant is named and supported. Arguments without a named warrant are flagged.

1. **Argumentation framework with dialectical tracker** (`lib/argumentation-framework.js`) — builds the argument-attack graph (rebuttal, undercutting, undermining) and identifies which arguments in the brief defeat which others. Pre-empts opposing-counsel rebuttals by surfacing the strongest counter-attack to each argument before the brief is filed.

1. **Multi-model peer review** (`lib/peer-review.js` + `lib/multi-model-v4.js`) — weighted-quorum vote across three flagship LLMs (Claude Opus 4.7 at 0.40 weight, Gemini 3 Pro at 0.35, GPT-5.2 Pro at 0.25). Used only when the deterministic + Bayesian path returns YELLOW; the LLMs may upgrade YELLOW → BLUE or YELLOW → GREEN with majority consensus, but they may **never** downgrade GREEN or upgrade RED/BLACK. The deterministic engine's veto is absolute.

The eight calculi run concurrently with `Promise.allSettled`; their outputs are reconciled by a per-argument synthesis function that applies the deterministic-first ordering rule and emits a single final cell.

---

5 · Epistemic status taxonomy

V4 does not collapse uncertainty to a single number. Each partial bit carries a named status with a documented reason (from `lib/advocacy-algorithm.js`):

Status	Bit value	Semantics
`VERIFIED`	1	Authority confirmed by verification pipeline.
`UNCITED`	0.5	Legal reasoning present but authority not directly cited.
`UNCERTAIN`	0.5	Semantic confidence too low for conclusive determination.
`PATTERN_ONLY`	0.5	Citation patterns detected but not verified.
`NO_RECORD`	0	No authority found in verification pipeline.
`CONTRADICTED`	0	Case contradicts the claimed holding.
`CANNOT_CHECK`	null	Verification unavailable — no record access.

The named status appears in the report. The user does not see "0.5"; the user sees "PATTERN_ONLY — three Pacific Reporter citations matched the pattern but none returned a verified case in our corpus or in CourtListener." That is the actionable fact.

---

6 · Enthymeme detection

`lib/enthymeme-detector.js` performs a second pass over the structured output from Gate B looking for implicit premises — the unstated assumptions an argument depends on. A standard syllogism has three propositions; an enthymeme has two stated and one suppressed. V4 surfaces every suppressed premise it can identify, flags whether the suppressed premise is itself supported by law or fact, and treats the suppressed premise as a fourth bit in the truth table when the suppression is material.

The `estimateEnthymemeGap()` function tells the user how much of the brief's argumentative weight rests on premises the brief never stated. A 40% gap is a brief that mostly persuades by what it does not say; V4 surfaces the implicit moves so the writer can either state them explicitly or strengthen the explicit ones.

---

7 · Why deterministic-first is the load-bearing rule

A commercial competitor could copy any single component above and still ship a tool that is unsafe to use, because the load-bearing rule is not any one calculus — it is the ordering rule between them.

Deterministic-first means: if the deterministic logic engine, given the structured output from Gate B, returns "invalid form" (modus ponens with denied antecedent, e.g.), then no Bayesian update from Dempster-Shafer, no posterior-shift from the Bayesian engine, no warrant-found from Toulmin, no consensus-GREEN from the three-model peer review, may move the cell back to VALID. The deterministic verdict is the floor. The probabilistic and neural layers may only refine within the bounds the deterministic verdict permits.

This is the opposite of every commercial competitor I am aware of. They all use the LLM as the final authority and the symbolic layer (if any) as a hint. I use the LLM as the hint and the symbolic layer as the authority. The result is a verifier that refuses to ship arguments the model would have shipped, and the refusal is reproducible — anyone with the same input gets the same output, because deterministic logic gives the same answer twice.

---

8 · Novelty claims

1. **Three-bit truth table over (Law × Fact × Logic)** as the dispositive classification of every legal argument, with nine named cells and a published color-flag taxonomy.
2. **Four-gate sequential decomposition pipeline** (presort, decomposition, verification, synthesis) where Gates A, C, and D run deterministically and only Gate B uses an LLM.
3. **Eight-parallel-calculi synthesis** (deterministic / Dempster-Shafer / Bayesian / defeasibility / burden / Toulmin / argumentation-framework / peer-review) where each calculus has a defined role and a defined override-precedence.
4. **Deterministic-first ordering rule** in which probabilistic and neural calculi may refine, but may never override, the deterministic verdict.
5. **Per-argument epistemic-status taxonomy** with seven named states (VERIFIED, UNCITED, UNCERTAIN, PATTERN_ONLY, NO_RECORD, CONTRADICTED, CANNOT_CHECK), each surfaced to the user with a documented reason.
6. **Enthymeme detection** with a gap estimator that quantifies the share of a brief's argumentative weight resting on unstated premises.

These six elements in conjunction are the V4 Algorithm. The repository at `lib/advocacy-algorithm.js` is the canonical source.
