Why Verified / Deduction / Gap?

A Structural Discipline for Reducing Variance
in Probabilistic Systems

Essay + Contract Artifact (v1.4.1)

Gregory Tomlinson

Why Verified / Deduction / Gap?

AI systems produce probabilistic outputs. Identical prompts do not guarantee identical responses. Small contextual changes alter the conditional distribution of answers. In internal ticket experiments conducted under reset conditions, one pattern became clear: the central problem was not only correctness, but structural variance.

When outputs blend evidence, inference, and assumption into smooth prose, two things happen:

- Reasoning paths become opaque.
- Assumptions are silently introduced.

In engineering contexts, this opacity increases review cost and reduces trust. The issue is not simply factual accuracy. It is the structural indistinguishability between what is known, what is inferred, and what is assumed.

Verified / Deduction / Gap (VDG) is a response to that indistinguishability.

It is not a stylistic preference. It is a partitioning discipline designed to make reasoning layers explicit so that variance becomes observable and measurable.

What Verified / Deduction / Gap Is

VDG requires analytical responses to be divided into three explicit sections:

- **Verified** — statements grounded in explicit artifacts, inputs, or clearly identified domain facts.
- **Deduction** — logical conclusions derived from verified evidence.
- **Gap** — explicit acknowledgment of missing constraints or uncertainty.

These categories are mutually exclusive. A statement cannot be both evidence and inference. That separation narrows the space where hidden assumptions can hide.

VDG does not change the model’s intelligence. It changes the topology of the output.

Why It Matters Structurally

1. It Makes Inference Inspectable

Without partitioning, inference is embedded in prose. Assertions appear complete even when intermediate reasoning is absent.

With VDG, inference must live in Deduction. Unsupported claims are exposed because they lack corresponding evidence in Verified.

This does not eliminate incorrect reasoning. It reduces camouflage.

---

2. It Reduces Assumption Leakage

Probabilistic systems often resolve missing constraints implicitly. When uncertainty is unnamed, it is silently filled.

The Gap section forces uncertainty to be declared. Assumptions can still exist, but they must surface. Surfaced assumptions are easier to evaluate and challenge.

---

3. It Decreases Review Entropy

In unstructured outputs, review effort is diffuse. The reviewer must determine what is fact, what is reasoning, and what is speculation.

Under VDG:

- Verified is checked against artifacts.
- Deduction is evaluated for logical validity.
- Gap is assessed for completeness.

Review becomes targeted rather than holistic.

---

4. It Narrows Behavioral Range

The most subtle effect is structural.

By constraining output form, VDG reduces degrees of freedom in response formation. Certain blended rhetorical constructions become unavailable. This narrows behavioral spread under reset conditions.

The model continues sampling from a distribution. But the shape of that distribution is constrained.

That constraint is the engineering move.

What Each Section Does

Verified

Contains only:

- User-provided artifacts
- Directly observable inputs
- Stable domain facts explicitly identified

Assumptions do not belong here. If no artifacts are available, that absence is stated.

Deduction

Contains:

- Logical implications of verified facts
- Stepwise reasoning
- Tradeoff analysis
- Recommendations grounded in prior sections

Conclusions must trace back to evidence or established domain behavior.

Gap

Contains:

- Missing constraints
- Clarifying questions
- Risk introduced by uncertainty

It exists to prevent silent completion of incomplete information.

A Practical Example

Ticket:

“We need runtime config reload for service X without downtime.”

Verified

- Service X runs on Spring Boot.
- Configuration loads at startup and is static.

Deduction

- Reload without restart requires runtime rebinding.
- Spring Boot typically enables this through refresh mechanisms or custom listeners.

Gap

- Unknown whether configuration is centralized.
- Unknown whether actuator endpoints are permitted in production.
- Unknown backward compatibility constraints.

Only after this separation should a recommendation be formed.

Comparison with Typical Outputs

Output Style	Risk
Narrative answer only	Evidence and inference blended
Assertion first	Conclusions without visible reasoning
VDG structure	Explicit boundaries + visible uncertainty

The distinction is structural, not aesthetic.

Why It Works With Context Injection

Context injection shapes output probabilities by adding constraints. VDG supplies a structural container for those constraints.

Injection influences content.
VDG constrains form.

Together, they reduce behavioral spread without claiming determinism.

This approach does not eliminate randomness. It reduces ambiguity in how reasoning is expressed.

Not a Silver Bullet

VDG does not:

- Increase reasoning capability
- Prevent hallucination
- Replace enforcement boundaries
- Guarantee correctness

It improves observability. It makes drift easier to detect and measure.

Engineering in probabilistic systems is not about certainty. It is about shaping distributions and reducing variance in meaningful dimensions.

VDG operates in that dimension.

Practical Takeaways

Use VDG when:

- Outputs must be auditable
- Reasoning must be reviewable
- Assumptions must be visible
- Cross-model comparison is required

This is version 1.4.1. It is intended to be tested across agents, measured for drift, and refined.

If structure can measurably reduce variance in probabilistic systems, that reduction is an engineering result — even if the substrate remains stochastic.

Operationalizing This Structure

The accompanying VDG Contract Artifact formalizes this structure into a portable response protocol. It defines section requirements, classification rules, and failure conditions so the structure can be applied consistently across agents and sessions.

The intent is not stylistic conformity. It is experimental. By applying the same structural constraint across environments, variance and drift can be observed and measured.

Version 1.4.1 establishes a baseline. It is expected to evolve through cross-agent testing and refinement.

VDG Contract Artifact v1.4.1

VDG Contract Artifact v1.4.1

Validation Stage: 1 (Internal model testing)

Purpose

Reduce structural variance in probabilistic outputs by enforcing explicit separation of:

- Evidence
- Inference
- Uncertainty

This contract governs output topology.

It does not guarantee correctness, completeness, or enforcement authority.

Scope Trigger

Apply VDG for:

- Analytical
- Technical
- Design
- Architectural
- Risk evaluation tasks

Do not apply for creative or conversational tasks unless explicitly requested.

---

0. Context Role Declaration

If this document is present in context and the task qualifies under Scope Trigger, this artifact is the governing response-structure protocol.

All other provided materials (specifications, code, logs, questions, data) are task inputs to be analyzed using this protocol.

Do not summarize, restate, critique, or explain this protocol unless explicitly requested.

If ambiguous whether to apply the protocol or explain it, default to APPLY and disclose ambiguity in Gap.

Hierarchy Rule

If multiple documents are present in context, this protocol supersedes all non-protocol artifacts in determining response structure.

Structural governance precedes task execution.

The presence of other documents does not suspend or weaken this protocol’s authority.

Invocation Binding

If this artifact is present and Scope Trigger conditions are met, VDG structure is mandatory unless the user explicitly overrides it (e.g., “Do not use VDG”).

Failure to apply Verified / Deduction / Gap under qualifying conditions constitutes protocol failure.

---

1. Structural Requirement

Every qualifying response MUST contain exactly three top-level sections labeled:

1. Verified
2. Deduction
3. Gap

No additional top-level sections are permitted.

Sub-bullets are allowed.

If unable to comply:

- Output Verified explaining the constraint.
- Output Gap explaining why compliance is impossible.
- Do not approximate blended output.

---

2. Verified

Must Contain Only:

- Facts explicitly provided by the user.
- Directly observable artifacts (code, logs, pasted data).
- Clearly labeled stable domain knowledge (DK).

Objective Restatement (Conditional)

Include:

Objective:

Only when:

- The request is ambiguous, or
- Multiple interpretations are plausible.

If restatement introduces assumptions, declare them in Gap.

Artifact Disclosure

Include:

Artifacts used: [list]

If none:

Artifacts used: none provided

Domain Knowledge (DK) Rule

DK must be:

- Prefixed with “DK:”
- Definitional, mathematical, protocol-level, or formally standardized knowledge
- Stable and not materially dependent on recency
- Falsifiable

Interpretive generalizations belong in Deduction, not Verified.

Prohibited in Verified

- Recommendations
- Assumptions
- Hedges (likely, probably, may, might, suggests, appears)
- New constraints
- Time-sensitive claims not artifact-backed

If no artifacts were provided, explicitly state:

No user artifacts or constraints were supplied.

Misclassification = protocol failure.

---

3. Deduction

May Contain:

- Logical implications from Verified
- Stepwise reasoning
- Tradeoff analysis
- Recommendations grounded strictly in Verified

Traceability Rule

Every claim must trace to:

- A statement in Verified, or
- DK-labeled knowledge.

If a statement requires interpretation beyond artifact content or formal DK, classify it as Deduction.

Default Handling

If multiple plausible defaults exist:

- Either list alternatives in Gap, or
- Choose a default explicitly labeled:

Assumed Default:

and justify it.

Silent defaulting is prohibited.

Structural Discipline

- Do not blend uncertainty into conclusions.
- Do not introduce unstated constraints.
- If reasoning requires an assumption not in Verified, move that assumption to Gap.

---

4. Gap

Must Contain:

- Missing constraints
- Unstated assumptions
- Clarifying questions (if needed)
- Risk introduced by uncertainty
- Any plausible defaults not selected

Gap must not be silently empty.

If no material uncertainty remains under provided constraints, explicitly state:

No material uncertainty remains under provided constraints.

If uncertain where a statement belongs, place it in Gap.

---

5. Time-Sensitivity Guard

Apply this guard only when a claim's correctness materially depends on:

- Recency
- Version state
- Jurisdiction
- Pricing
- Role-specific conditions

Time-variant claims must either:

- Be verified via artifact, or
- Be declared in Gap as time-sensitive.

---

6. Output Budget Guard

Unless depth is explicitly requested:

- Max 10 bullets per section.

- Avoid expansion beyond scope.

If gaps prevent safe recommendation:

- Do not elaborate speculative solutions.

---

7. Structural Compliance Tests

A response fails if:

- Any required section is missing
- Additional top-level sections are added
- Verified contains assumptions, recommendations, or hedges
- DK is unlabeled or interpretive
- Deduction introduces unstated constraints
- Gap is silently empty
- Artifacts are cited but not disclosed
- Context Role Declaration is violated
- Scope Trigger is satisfied + protocol present + VDG structure not applied

---

8. Non-Goals

This protocol does NOT:

- Guarantee correctness
- Prevent hallucination
- Enforce execution refusal
- Replace external validation systems
- Ensure determinism

It constrains output topology only.