Why Verified / Deduction / Gap? v1.4.2

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 1.4.2. 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.

VDG Contract Artifact v1.4.2.pdf establishes a baseline. It is expected to evolve through cross-agent testing and refinement.