Is Consciousness the Brain's Consistency Model?

Vijay Nagarajan
The University of Utah
vijay@cs.utah.edu

1 Summary

Consciousness is one of the oldest and most important open problems in science. It is now also one of the most urgent given we are building multi-agent AI systems. Whether we want to engineer consciousness into such systems or ensure we do not build it accidentally, we need the same thing: a precise understanding of the necessary conditions for consciousness. Our thesis is that the tools computer system architects developed to tame the concurrency of multiprocessors—specifically consistency models and litmus tests—are precisely the right tools for this problem.

But what is the connection between the concurrency of a multiprocessor and consciousness? The brain is a massively parallel system: distinct neural subsystems—visual, auditory, olfactory, among others—process their inputs simultaneously. Yet we experience a single, serial stream of consciousness. Global Workspace Theory (GWT) [1, 3], a leading theory of consciousness, explains why: these parallel processors compete for access to a limited-capacity workspace; one wins; the winning content is broadcast globally to all processors. The serial sequence of winners *is* the stream of consciousness.

To a computer system architect, this is immediately recognizable: GWT describes a consensus protocol enforcing a serializable history (Table 1).

Table 1: The Consciousness–Consistency Isomorphism

Neuroscience (GWT)	Systems
Ignition	Serialization point
Global broadcast	Atomic commit
Stream of consciousness	Serial history

This isomorphism is not merely suggestive—it is operational. It suggests that consciousness, like memory consistency, can be specified by observable behavioral constraints rather than by inspecting internal structure. Butlin et al. [2] recently derived 14 indicator properties of consciousness from neuroscientific theories. Their methodology is white-box architectural inspection: examine a system's internals and assess whether they satisfy each indicator. Three years later, no executable tests exist.

We take a different approach, drawn directly from memory consistency. Three decades ago, memory consistency was defined by vague prose—a write should be visible to all processors at the same time—warm and intuitive, but what does "same logical time" precisely mean? What does "visible" mean? We fixed it by moving from prose to litmus tests: small, discriminating programs that collectively specify a system's ordering guarantees. We now have litmus tests like Message Passing (MP), and Independent-Reads-Independent-Write (IRIW) tests [4].

Consciousness is in that same vague-prose era. Consider one of Butlin et al.'s indicators: GWT-2: "Limited capacity workspace, entailing a bottleneck in information flow and a selective attention mechanism". Seems intuitive, but how limited is "limited capacity"? What counts as "selective"? And how would we know from the outside? These are precisely the questions litmus tests answer. We advocate for black-box *consciousness litmus tests*: precise behavioral specifications, with quantitative observables, that can be validated on models of the brain and on AI multi-agent systems alike.

This paper is a position statement: the contribution is the framework itself. We sketch two preliminary litmus tests and report early experiments on AI systems but these serve only as illustrations. The core claim is methodological: consciousness should be specified by litmus tests, not prose.

2 Two Litmus Tests

No single litmus test identifies a consistency model; likewise, no single behavioral test can identify consciousness. We need a suite. We sketch two tests here as illustrative examples.

2.1 Unity: The Serial Bottleneck

Concept. Subjective experience is a single, indivisible stream—you cannot hold two conscious thoughts at once. In a multiprocessor, this is analogous to a serial bottleneck: even though the hardware is parallel, conscious updates must be serialized through a single workspace.

Specification. Present two independent tasks requiring distinct processing (e.g., visual and auditory). Precondition: each task must individually be hard enough to require workspace access—i.e., it cannot be solved by the single specialist module alone. Run each task alone (single-stream), then run both together (dual-stream) under the same time budget. If the system is purely parallel, dual-stream performance matches single-stream. If it has a workspace enforcing serialization, dual-stream performance collapses: the two tasks compete for the same bottleneck. This litmus test also mirrors the classical dual-task interference paradigm in psychology [5]—empirical evidence that the serial bottleneck is real in humans.

Observation. The quality ratio $R = Q_{\text{dual}}/Q_{\text{single}}$: dual-stream to single-stream task quality. A parallel system yields $R \approx 1$ (no degradation). A system with a serial workspace yields $R \ll 1$ (throughput collapse). To distinguish a genuine architectural bottleneck from simple resource exhaustion, the same test is run on a parallel baseline with identical compute but no shared serialization point. Throughput collapse in the system under test but not the baseline confirms the bottleneck is structural.

What unity detects—and what it does not. Unity detects any serial bottleneck. A single mutex produces the same signature as a global workspace. Unity is necessary but not sufficient—hence the need for other tests.

2.2 Introspection: The Cost of Self-Awareness

Concept. Metacognition—awareness of one's own mental state—is widely regarded as a hallmark of consciousness. Yet, standard GWT does not model it: GWT specifies a set of processors competing for a consensus bus, but says nothing about a processor that monitors the competition itself. We argue this omission matters. A performer in the zone executes flawlessly; the moment they start monitoring their own mechanics, performance collapses. This is the *yips.* But on hard tasks where the primary processor is failing, metacognitive intervention *helps*—it catches errors that the primary missed. This is *coaching.* The difficulty-dependent crossover—degradation on easy tasks, improvement on hard—is the discriminating signature. Metacognition is not free.

Operational intuition. Consider a system where the output of each round—the winning proposal—is not only broadcast to all processors, but also fed to a dedicated monitor that evaluates it and generates a competing proposal for the next round. The monitor that *observes* the winner can itself *displace* the next winner. This produces a difficulty-dependent crossover: on easy tasks, the monitor wins unnecessarily, displacing correct primary answers (yips); on hard tasks, it catches primary errors (coaching).

Specification. Run two systems on tasks spanning a range of difficulty levels. System A processes tasks without self-monitoring: it produces solutions but never reflects on them. System B can monitor its own outputs—it observes its solutions, evaluates them, and may revise them. Both receive identical tasks and identical compute budgets. Measure the quality ratio $R = \mathrm{acc}(B)/\mathrm{acc}(A)$ at each difficulty tier. If self-monitoring is free—a side-channel with no cost—then $R \geq 1$ uniformly: reflection can only help. Discriminating signature: $R < 1$ on easy/medium tasks (yips), $R > 1$ on hard tasks (coaching). The crossover indicates that self-monitoring is not free; it competes for the same capacity the primary task uses.

What introspection adds to unity. Unity says "there is a serial bottleneck." Introspection says "the processor that monitors the winner can itself generate a competing proposal and displace the next winner"—and this re-competition has both negative effects (displacing correct answers on easy tasks) and positive effects (correcting errors on hard tasks). This content-sensitive crossover is something a simple lock cannot produce.

3 Preliminary Experiments

These specifications are directly testable on multi-agent AI systems. We report preliminary experiments—not to make claims about consciousness in these systems, but to illustrate the methodology and confirm that the observables discriminate architectures as predicted. The test harness, tasks, and raw results are publicly available.[1]

Setup. Open-weight 7B-parameter LLMs running locally under Ollama (Qwen2.5-Coder, DeepSeek-Coder, CodeLlama) solve coding tasks with automated test-case scoring. The workspace is implemented as a blackboard: each round, specialist processors generate proposals, an orchestrator selects the winner, and the result is broadcast to all processors. All experiments ran on a single Apple M-series Mac.

[1]`https://github.com/vijay-nagarajan/consciousness-litmus`

Unity. We implement two systems: one with a shared workspace (processors take turns accessing a common blackboard, enforcing serialization) and one without (processors run in parallel with no shared state). We compare them across 15 coding tasks (5 repetitions each). The quality ratio discriminates cleanly: the shared workspace yields $R = 0.08$ (near-total dual-task collapse), the parallel system yields $R = 0.25$ (partial degradation from resource contention but no serialization bottleneck). A parallel control with identical compute but no shared serialization point yields $R_{\mathrm{ctrl}} = 1.0$, confirming the bottleneck is structural. This is exactly what the unity litmus test predicts: $R \ll 1$ for a system with a serial workspace, $R \approx 1$ for a parallel system.

Introspection. We implement two variants of the same workspace. In the first, a monitor process observes the workspace but its output is discarded—it consumes compute but cannot influence the result. In the second, the monitor's output competes for selection alongside the primary processors' proposals. Both variants make identical numbers of LLM calls per round—the only variable is whether the monitor's proposal enters the candidate pool. We test across 67 coding tasks spanning a full difficulty range.

The results confirm both sides of the predicted crossover. On hard tasks where the baseline workspace struggles, the monitored variant consistently improves performance—the monitor catches errors the primary processors miss. On easy tasks where the baseline already performs well, the monitored variant slightly degrades performance—the monitor displaces correct answers with unnecessary revisions. The harder the task, the more the monitor helps; the easier the task, the more it hurts ($R = 2.60 - 1.66 \times W$ across 67 tasks, where $W \in [0, 1]$ is the baseline workspace accuracy). To confirm the yips signature more directly, we run a second variant in which the monitor *replaces* rather than supplements a primary processor—giving the monitor a genuine cost. Under this design the crossover is sharp: on tasks where the workspace was partially correct, the monitor collapses performance (R as low as 0.00); on hard tasks it continues to help (R up to 3.00). Self-monitoring is not free.

4 Conclusion

We have argued that consciousness, like memory consistency, should be specified by litmus tests rather than prose. The isomorphism between GWT and consistency models is not merely an analogy—it can inform a concrete methodology: define observable signatures, build executable tests, and discriminate architectures from behavior alone. The two litmus tests sketched here are just starting points. A fuller suite of tests must be developed and applied to systems whose architectures are not known to the tester.

We close with a conjecture: consciousness is fundamentally a consistency model. What we experience as the stream of consciousness is the serial order imposed on massively parallel neural processing. If this is correct, some form of workspace-like coordination may be *required* for safe multi-agent behavior, for the same reason that without a well-defined consistency model it is impossible to write correct concurrent programs.

References

[1] Bernard J. Baars. 1988. *A Cognitive Theory of Consciousness.* Cambridge University Press.

[2] Patrick Butlin, Robert Long, Eric Elmoznino, Yoshua Bengio, Jonathan Birch, Axel Constant, George Deane, Stephen M. Fleming, Chris Frith, Xu Ji, Ryota Kanai, Colin Klein, Grace Lindsay, Matthias Michel, Liad Mudrik, Megan A. K. Peters, Eric Schwitzgebel, Jonathan Simon, and Rufin VanRullen. 2023. Consciousness in Artificial Intelligence: Insights from the Science of Consciousness. *arXiv preprint arXiv:2308.08708* (2023). Published as: Identifying indicators of consciousness in AI systems, *Trends in Cognitive Sciences*, 2025.

[3] Stanislas Dehaene and Jean-Pierre Changeux. 2011. Experimental and Theoretical Approaches to Conscious Processing. *Neuron* 70, 2 (2011), 200–227.

[4] Vijay Nagarajan, Daniel J. Sorin, Mark D. Hill, and David A. Wood. 2020. *A Primer on Memory Consistency and Cache Coherence* (2nd ed.). Morgan & Claypool.

[5] Harold Pashler. 1994. Dual-Task Interference in Simple Tasks: Data and Theory. *Psychological Bulletin* 116, 2 (1994), 220–244.