Beyond Token Time

Representation Beyond Tokens, Fixed Clocks, and Modality Boundaries

Position Paper and Research Proposal

Nako Sung
NAVER Cloud, Hyperscale AI

Version 1.2 – revised September 2, 2026

Abstract. Modern foundation models operate in high-dimensional continuous representation spaces, but their computation is repeatedly constrained by narrow interfaces: discrete tokens, a single serial stream, fixed or globally shared clocks, and modality-specific boundaries. These constraints were productive scaffolds for scalable training, not necessarily the native form of intelligent computation. This paper proposes a unifying view: many apparent architectural limitations arise from projecting a rich internal state into interface coordinates too early. Tokens quantize; clocks synchronize; modalities partition; serial streams order. Each can force premature commitment. We therefore separate *representation* from *serialization*, *schedule*, and *modality*. A Persistent Multi-Stream Latent World State maintains several evolving components, while modality adapters act as lenses and computation is triggered by relevant events. Concrete mechanisms—including null-token continuous recurrence and gated latent carry—are presented as minimal ways to relax specific bottlenecks rather than as the central thesis. We close with compute-matched experiments that independently test the value of relaxing discrete, temporal, modal, and serial constraints, while treating monitorability as a first-class cost of representational freedom.

1 The Interface Is Narrower Than the Representation

A transformer layer maps activations through a high-dimensional continuous space. Multiple features, hypotheses, entities, plans, and cross-modal correlations can coexist in that state. There is no architectural requirement that one internal vector correspond to one word, one modality, one object, or one unit of time.

Yet the state must repeatedly cross narrow boundaries. In ordinary autoregressive generation, a hidden state h_t is projected into a vocabulary distribution, a discrete token y_t is selected, embedded, and fed back:

$$h_t \to y_t \to E(y_t) \to h_{t+1}. \tag{1}$$

This is an extraordinarily successful training scaffold. The question is not whether tokenization was a mistake. The question is whether the scaffold that made intelligence trainable must remain the native form in which intelligence computes.

Central thesis. The bottleneck may lie less in what a model can represent than in the interfaces through which we force that representation to evolve.

The phrase *Beyond Token Time* therefore refers to more than silent reasoning. "Token" denotes the broader act of collapsing internal state into a discrete interface symbol; "time" denotes the assumption that all computation must advance on one shared step counter.

2 One Pattern, Four Bottlenecks

Four familiar constraints can be viewed as variants of the same operation: projection of a broad internal state into interface coordinates.

Discrete token bottleneck. A continuous state is quantized into one vocabulary symbol. This forces semantic commitment and discards distinctions not useful for the chosen token.

Temporal bottleneck. Additional computation is tied to token position, fixed depth, or another globally shared cadence. Easy and difficult states receive the same nominal schedule unless extra machinery is added.

Modality bottleneck. Text, image, audio, and sensor data enter through distinct interfaces and are often treated as distinct internal modes. Practical adapters are necessary, but their boundaries need not define the ontology of internal state.

Seriality bottleneck. Several entities, hypotheses, goals, and perceptual processes may coexist, yet they are often serialized into one sequence or compressed into one entangled state updated at one rate.

These constraints share a consequence: *premature commitment*. A system commits early to a symbol, an order, a modality partition, or one update clock even when the underlying state could remain unresolved.

$$\text{representation} \neq \text{serialization} \neq \text{schedule} \neq \text{modality}. \tag{2}$$

3 Tokens: A Useful Quantizer, Not Necessarily the Only State Carrier

Explicit chain-of-thought (CoT) can be interpreted as a particularly stable form of adaptive recurrent computation:

$$h_t \to y_t^{\text{reason}} \to E(y_t^{\text{reason}}) \to h_{t+1}. \tag{3}$$

The number of reasoning steps is expressed by trace length rather than fixed architecture depth. Expert traces make this easy to bootstrap with supervised learning. This helps explain why CoT was such an effective route to variable compute.

The same mechanism imposes a discrete semantic bottleneck. The recurrent state passed between steps must be representable through language tokens. Several hypotheses may coexist internally, but the feedback path serializes one commitment at a time.

The problem is exclusivity, not tokens. Tokens are excellent communication and supervision interfaces. The architectural question is whether they should be the only recurrent feedback channel.

4 Time: Token Time, Compute Time, and World Time

Current systems often conflate three different clocks:

$$t_{\text{token}} \neq t_{\text{compute}} \neq t_{\text{world}}. \tag{4}$$

Token time advances when another visible symbol is emitted. **Compute time** advances when internal state receives another update. **World time** advances when relevant external events occur.

There is no general reason these clocks should coincide. A difficult problem may deserve several model evaluations without advancing visible output. During full-duplex interaction, speech can continue while audio arrives, vision updates at another cadence, and a tool result arrives asynchronously.

This motivates event-driven computation. Let events $e_1, e_2, \ldots$ activate only the state components they affect. An internal reasoning trigger can request more compute; an acoustic event can update speaker and semantic state; a visual event can update entity and control state. The scheduling unit becomes relevance rather than token production.

5 Modality as a Lens

Let s denote an underlying world state. An observation through modality m can be written

$$o^{(m)} = g_m(s) + \epsilon^{(m)}, \tag{5}$$

where g_m captures modality-specific measurement and $\epsilon^{(m)}$ captures noise and loss.

An RGB camera samples selected wavelengths at a chosen spatial and temporal resolution. A microphone samples pressure changes. Korean and English encode overlapping events through different symbol systems. From this perspective, modalities are not necessarily separate kinds of internal thought; they are different *lenses* on an underlying state.

Modality-specific encoders, tokenizers, and decoders remain useful engineering interfaces. The stronger claim is only that internal representation need not inherit those interface boundaries. One object's appearance, name, sound, motion, and affordances may be coupled properties of one persistent state.

Modality-agnostic does not mean adapter-agnostic. Sensors and outputs may use specialized adapters while sharing an internal world representation that is not owned by any one modality.

6 Seriality: One Output Stream Is Not One Internal Process

A video contains many independently evolving entities while being serialized on one timeline. A dialogue scene simultaneously preserves actor identity, voice, gaze, turn-taking, camera motion, and shared context. A textual explanation may say "first consider A, then B," while evidence for both hypotheses can coexist internally.

Thus,

$$\text{one output stream} \neq \text{one internal process}. \tag{6}$$

This motivates persistent multi-stream latent state. At event e_n, maintain

$$Z_n = [z_n^{(1)}, z_n^{(2)}, \ldots, z_n^{(K)}], \tag{7}$$

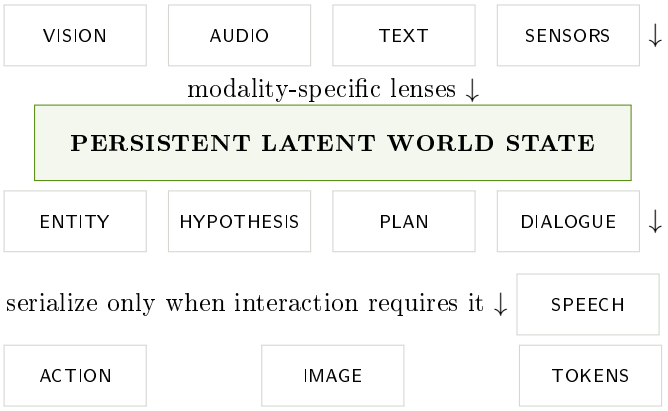
where components may correspond to entities, hypotheses, plans, dialogue state, memory items, perceptual processes, or predicted futures. A routing policy chooses which components to update:

$$a_n = \pi_{\text{route}}(Z_n, e_n), \qquad a_n \in \{0,1\}^K, \tag{8}$$

$$z_{n+1}^{(k)} = \begin{cases} F_k(z_n^{(k)}, e_n, Z_n), & a_n^{(k)} = 1, \\ z_n^{(k)}, & a_n^{(k)} = 0. \end{cases} \tag{9}$$

7 A Unified Design Principle

The common principle is to preserve internal state and project only when needed.



Tokens become one readout interface rather than the unique format of state. Modalities become observation and output lenses rather than ownership boundaries. Compute scheduling can move independently from visible token production. Persistent state components can update selectively rather than synchronously.

Design principle. Tokens are interfaces. Modalities are lenses. Time is a schedule. None of them has to be the native format of thought.

8 Concrete Relaxation I: Null-Token Continuous Recurrence

The null-token mechanism introduced in v1.1 is best understood as one local relaxation of the broader thesis. It targets two bottlenecks at once: discrete recurrent feedback and the coupling between token time and compute time.

Introduce a reserved internal action $\varnothing$. Let

$$z_t = P(h_t), \qquad g_t = \sigma(W_g h_t). \tag{10}$$

Then define the next input embedding as

$$x_{t+1} = E(y_t) + \mathbf{1}[y_t = \varnothing] g_t \odot z_t. \tag{11}$$

When y_t is visible, decoding proceeds normally. When $y_t = \varnothing$, the runtime suppresses serialization and performs another transformer pass using the null embedding plus gated continuous carry.

A fixed number of silent passes is close to tied-weight depth expansion. The more interesting regime is variable recurrence, where the model learns how much compute a state requires. Reinforcement learning is a natural way to learn this policy because the correct number of silent steps is usually not uniquely supervised, although distillation and curriculum methods are also possible.

9 Concrete Relaxation II: Gated Multi-Stream Carry

The carry need not be one vector. With persistent state $Z_t = [z_t^{(1)}, \ldots, z_t^{(K)}]$, a controller can form

$$x_{t+1} = E(y_t) + \sum_{k=1}^{K} g_t^{(k)} \odot P_k(z_t^{(k)}). \tag{12}$$

This allows several state components to influence the next computation without first being serialized into one textual chain. A visible token can still anchor interaction while continuous channels preserve unresolved or cross-modal information.

The mechanism is intentionally modest: it does not prove that explicit slots are always better than entangled hidden states. It provides a falsifiable way to test whether forcing all recurrent information through one discrete stream is unnecessarily restrictive.

10 Relation to Recurrent-Depth Reporting and Astra

Public reporting describes OpenAI's Astra as using recurrent depth or a looped transformer, while OpenAI has not publicly disclosed the exact mechanism. The null-token architecture above should therefore be treated as an independent hypothesis, not an attribution. It shows one minimal way to obtain functionally similar variable unrolled computation while preserving an ordinary decoder loop.

This distinction matters because recurrent depth alone does not explain the broader argument of this paper. Even a perfect recurrent-depth mechanism would address mainly the temporal and token bottlenecks; modality and seriality remain separate questions.

11 Monitorability as the Cost of Representational Freedom

Text CoT has a special property: computation and a human-readable trace partially share the same channel. If more reasoning requires more textual steps, the architecture leaves observable checkpoints almost automatically.

Once state can evolve without serialization, this alignment disappears. A richer latent state may improve efficiency or capability while making the internal trajectory less directly legible. Monitorability therefore becomes an explicit objective rather than a free by-product of the interface bottleneck.

Possible tools include auxiliary decoders, probes, sparse verbal checkpoints, causal interventions, latent-state per-

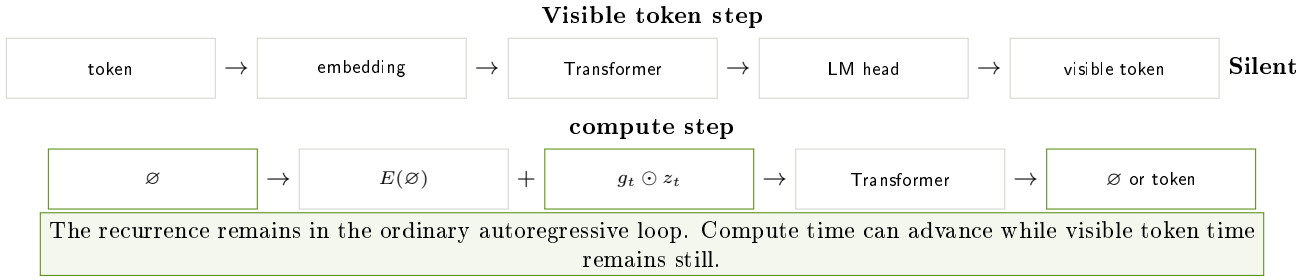


Figure 1: Null-token continuous recurrence as one concrete relaxation of token and timing bottlenecks.

turbation tests, and consistency checks across modality readouts. The goal is not necessarily to recover a verbatim hidden transcript; it is to preserve enough observability to detect goal shifts, deception, unsafe planning, or unstable recurrent dynamics.

General trade-off. Relaxing interface bottlenecks increases representational freedom. Monitoring must then be deliberately engineered rather than inherited from serialization.

12 Experimental Program

The central hypothesis should be decomposed into independent bottleneck tests.

12.1 Discrete bottleneck

Compare explicit CoT with null-token latent carry under matched transformer FLOPs. Remove continuous carry while preserving repeated depth to determine whether gains come from richer feedback or merely extra compute.

12.2 Temporal bottleneck

Compare fixed recurrent depth with learned variable recurrence. Test whether compute allocation correlates with task difficulty and improves the accuracy–latency frontier.

12.3 Modality bottleneck

Present the same underlying event through text, image, audio, and combinations thereof. Measure cross-modal consistency of shared state, transfer to non-linguistic action, and degradation when modality-specific intermediate serialization is forced.

12.4 Seriality bottleneck

Use environments with independently evolving entities and delayed evidence. Compare a single entangled state against explicit persistent components under matched compute. Measure interference, retention, revision cost, and interruption handling.

12.5 Monitorability cost

As representational freedom increases, measure probe decodability, intervention predictability, sparse-checkpoint fidelity, and whether unsafe internal changes can be detected before output.

Killer test. If relaxing token, time, modality, or seriality bottlenecks fails to improve accuracy, efficiency, robustness, or interruption handling under matched compute, the additional latent architecture is not justified.

13 Discussion

The success of language models makes it easy to confuse a powerful interface with a native substrate of intelligence. Tokens are exceptionally useful because they provide discrete supervision, scalable datasets, and clear communication. Fixed steps simplify optimization and systems engineering. Modality-specific adapters exploit structure. Serial streams make outputs easy to evaluate.

The argument here is not to discard these interfaces. It is to stop assuming that their boundaries must also define the boundaries of internal computation.

A model's continuous state is already richer than its emitted token. A multimodal model already integrates signals that do not share one physical representation. A full-duplex agent already encounters events that do not respect one output clock. A world model already contains several things at once. The architectural opportunity is to let these facts persist across the computation boundary instead of repeatedly collapsing them into interface coordinates.

14 Conclusion

Beyond Token Time is a proposal about architectural separation. Representation can remain continuous while

readout is discrete. Compute time can diverge from token time. Modalities can remain specialized at the edges while sharing an internal world state. Several processes can coexist while one interface stream is serialized for communication.

Null-token recurrence, gated latent carry, and persistent event-driven streams are concrete mechanisms for testing this view, not the view itself.

Preserve representation. Compute when needed. Project only when an interface asks.

Revision History

v1.2 – September 2, 2026. Reframed the central thesis around representation-space versus interface bottlenecks: discrete tokens, fixed timing, modality boundaries, and serial streams. Repositioned null-token recurrence as one concrete relaxation mechanism rather than the center of the argument.

v1.1 – September 2, 2026. Added null-token plus gated continuous recurrence, adaptive loop count, serving implications, Astra comparison, and monitorability discussion.

v1.0 – August 21, 2026. Initial persistent multi-stream latent world state, modalities-as-lenses, event time, observer-relative ordering, and frame-conditioned readout proposal.

References

[1] S. Hao et al., "Training Large Language Models to Reason in a Continuous Latent Space," arXiv:2412.06769, 2024.

[2] A. Jaegle et al., "Perceiver IO: A General Architecture for Structured Inputs and Outputs," ICLR, 2022.

[3] Z. Yue et al., "Hybrid Latent Reasoning via Reinforcement Learning," NeurIPS, 2025.

[4] R. You et al., "Parallel Test-Time Scaling for Latent Reasoning Models," ACL, 2026.

[5] G. Su et al., "Multi-Stream LLMs: Unblocking Language Models with Parallel Streams of Thoughts, Inputs and Outputs," arXiv:2605.12460, 2026.

[6] H. Kohli et al., "Loop, Think, & Generalize: Implicit Reasoning in Recurrent-Depth Transformers," arXiv:2604.07822, 2026.

[7] S. Li et al., "DeepLoop: Depth Scaling for Looped Transformers," arXiv:2607.13491, 2026.

[8] Y. Fan et al., "Bridging the Gap Between Latent and Explicit Reasoning with Looped Transformers," arXiv:2606.31779, 2026.

[9] A. Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.

[10] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," CACM, 1978.

[11] Z. Kong et al., "Let Them Talk: Audio-Driven Multi-Person Conversational Video Generation," arXiv:2505.22647, 2025.

[12] OpenAI, "Sora 2 System Card," September 2025.