Beyond Token Time

Persistent Multi-Stream Latent World States for Modality-Agnostic Intelligence

Position Paper and Research Proposal

Nako Sung
NAVER Cloud, Hyperscale AI

Version 1.1 – revised September 2, 2026

Abstract. Modern language models maintain rich continuous hidden states, yet their prevailing operating regime repeatedly projects those states through a discrete token interface and advances on a single token clock. Continuous latent reasoning, multi-stream language modeling, and recurrent-depth transformers each relax part of this constraint. This revision adds a concrete bridge from token reasoning to continuous recurrence: reserve a silent null token; when it is selected, suppress external serialization and feed the next transformer pass with the null-token embedding plus a gated continuous vector from the previous state. Repeated null emissions create variable unrolled depth along the ordinary autoregressive decode loop, without requiring a separate vertical recurrent stack. We then place this primitive inside a broader architectural hypothesis: intelligent computation can be organized around persistent structured state, selective event-triggered updates, and modality-specific readout rather than a single global token sequence. Public reporting about OpenAI's Astra motivates a comparison with recurrent depth, but the null-token mechanism proposed here is an independent architectural hypothesis, not an attribution of Astra's implementation. The paper closes with compute-matched experiments designed to falsify both the recurrence mechanism and the broader Persistent Multi-Stream Latent World State proposal.

1 Introduction

Autoregressive language models follow a simple rhythm. Given a token prefix $x_{\leq t}$, a transformer computes a continuous hidden representation h_t, maps it to a vocabulary distribution, selects a discrete symbol, and repeats:

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

This rhythm is well suited to language generation. Whether internal computation must follow the same rhythm is less clear. Coconut weakens that coupling by feeding a hidden state back as a subsequent input embedding rather than decoding every intermediate state into a token [1]. Removing intermediate text, however, addresses only one constraint. Computation is still commonly framed as a single sequence advancing under a global step index.

Interactive environments are not synchronized this way. A conversational agent may listen, update beliefs, plan, speak, and detect interruptions concurrently. A robot may update control at high frequency while invoking planning only when needed. A scene contains many entities whose states evolve at different rates.

Central question. What if a token were only an interface action, while additional computation could continue in continuous state until an event or interface requires commitment?

The contribution lies not in latent reasoning, recurrent depth, parallel streams, multimodality, or reinforcement learning alone, but in a concrete interface primitive and its architectural combination with persistent event-driven state.

2 Chain-of-Thought as Discrete Recurrence

Explicit chain-of-thought (CoT) can be viewed as a stable form of adaptive recurrent computation. The model emits an intermediate reasoning token, re-embeds that token, invokes the same transformer again, and continues for a variable number of steps. The recurrent loop is therefore externalized through a discrete semantic channel:

$$h_t \to y_t^{\text{reason}} \to E(y_t^{\text{reason}}) \to h_{t+1}. \tag{2}$$

This perspective helps explain why CoT was such an effective route to adaptive computation. Supervised expert traces provide a variable-length training target without requiring the model to invent an unobserved halting policy from scratch. The cost is a discrete bottleneck: every intermediate state that participates in the loop must be compressed into a token or short textual fragment before being fed back.

Interpretation. CoT is a recurrent loop whose state transition is forced through a human-readable discrete bottleneck.

Once a model already possesses strong reasoning behavior, a natural next step is not necessarily to discard the autoregressive loop. It is to relax the bottleneck inside that loop.

3 Null-Token Continuous Recurrence

Introduce a reserved internal action $\varnothing$ (a silent or null token). Let y_t be the action selected by the language-model head, $E(y_t)$ its token embedding, and let a projection of the current hidden state define a continuous carry vector

$$z_t = P(h_t), \qquad g_t = \sigma(W_g h_t). \tag{3}$$

The next input embedding is

$$x_{t+1} = E(y_t) + \mathbf{1}[y_t = \varnothing]\, g_t \odot z_t. \tag{4}$$

If y_t is an ordinary token, the model behaves conventionally. If $y_t = \varnothing$, the runtime suppresses that action from the user-visible stream and performs another transformer pass using the null embedding plus gated continuous carry. The language-model head then chooses again between another silent step and a visible token.

Repeated null emissions increase *unrolled* computation depth along the generation axis. A fixed number of silent passes is functionally close to a tied-weight depth expansion. The more interesting regime is variable recurrence, in which the model learns how much additional computation a particular state requires. Loop count becomes a policy decision, resembling adaptive computation time but implemented through the existing autoregressive scheduler.

3.1 Always-on gated carry

Equation 4 is the minimal design because continuous carry is activated only after a null action. A more general form keeps the side channel available at every step:

$$x_{t+1} = E(y_t) + g_t \odot z_t, \tag{5}$$

with training free to drive g_t toward zero when the discrete token is sufficient. The token then serves as a semantic anchor and interface action rather than the sole carrier of recurrent state.

3.2 From one carry vector to many streams

Let persistent state contain K components $Z_t = [z_t^{(1)}, \ldots, z_t^{(K)}]$. A controller may mix them into the next computation:

$$x_{t+1} = E(y_t) + \sum_{k=1}^{K} g_t^{(k)} \odot P_k(z_t^{(k)}). \tag{6}$$

This is the bridge from continuous recurrence to the broader multi-stream proposal. A single textual CoT serializes one sequence of commitments. Multiple continuous components can simultaneously preserve a visual hypothesis, linguistic interpretation, plan, candidate explanations, sensor state, or other internal processes before any one of them must become text.

3.3 Training the halting policy

Reinforcement learning is a natural stage for variable recurrence because there is generally no unique target latent trajectory or correct number of silent passes. One can reward task success while charging for excess recurrent compute:

$$R = R_{\text{task}} - \lambda_n N_{\varnothing} - \lambda_s C_{\text{stability}} - \lambda_m C_{\text{monitor}}. \tag{7}$$

This does not imply that RL is mathematically required. Distillation, latent supervision, or curriculum methods can be used when targets for intermediate states are available. The practical attraction of outcome-driven optimization is that it avoids labeling an invisible trace.

3.4 Serving implications

The API surface can be small: a reserved internal action and an option enabling continuous carry. The runtime change is more substantial. Each active sequence must retain latent side state; the decode loop must accept dynamically constructed input embeddings; silent steps must still update KV cache consistently; batching and scheduling must account for variable numbers of internal passes; and the sampler must prevent null actions from entering the user-visible output stream. Conceptual simplicity should therefore not be confused with a one-line inference-engine patch.

4 Astra as a Comparison, Not an Attribution

The Information reported that OpenAI's Astra uses "recurrent depth" or a "looped transformer" and can repeatedly process information before emitting the next word; the report also states that OpenAI limits the technique so a legible CoT remains available [9]. OpenAI's own public material states that Astra will be deployed with additional chain-of-thought monitoring, but does not disclose the null-token mechanism proposed here [10].

OpenAI Chief Scientist Jakub Pachocki subsequently stated that the computation-graph depth of OpenAI's present frontier models, including Astra, is within a factor of two of GPT-4, and that the negative trend in CoT monitorability is driven by reasons not contingent on architecture changes [11]. This is important evidence against treating architecture reporting as a complete explanation of monitorability.

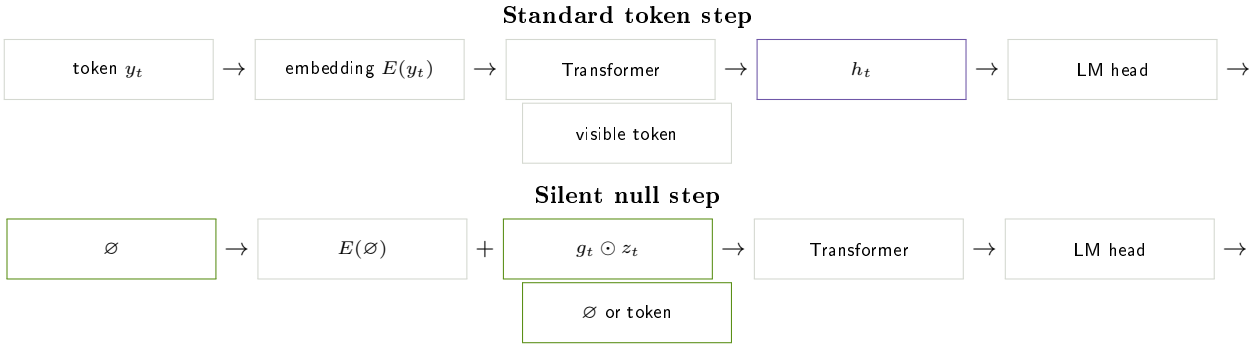


Figure 1: Proposed null-token continuous recurrence. The null action advances compute time while visible token time need not advance.

Boundary of the claim. The null-token plus gated-latent recurrence in this paper is our proposed mechanism. It is one minimal way to realize functionally similar unrolled recurrent computation while preserving the ordinary token-generation loop. Public information does not establish that Astra uses this exact mechanism.

Looped and recurrent-depth transformers are themselves an active research direction. Recent studies show that repeated tied-depth computation can support implicit compositional reasoning, depth extrapolation, and parameter-efficient depth scaling [6, 7]; LOTUS further studies looped transformers as a bridge between latent and explicit reasoning [8]. The proposed null action differs primarily in *where the recurrence is controlled*: the decode policy itself decides whether to spend another continuous step before serialization.

5 Monitorability after the Discrete Bottleneck

Text CoT is unusual because computation and a human-readable trace are partially aligned: extra reasoning steps tend to leave extra visible symbols. Once multiple transformer passes can occur behind null actions or continuous side state, that alignment is no longer structural. A visible answer may expose only sparse checkpoints of a richer latent trajectory.

This does not imply that latent recurrence is the sole cause of declining monitorability. The public OpenAI statement explicitly warns against that conclusion [11]. The narrower implication is architectural: *readability is no longer an automatic by-product of the computation loop.* If oversight matters, it must become an explicit research objective through auxiliary decoders, probes, sparse verbal checkpoints, causal interventions on latent state, or other monitoring methods.

From explainability by serialization to monitorability by design. A model that can think without speaking may still be monitorable, but the monitoring channel has to be deliberately preserved or reconstructed.

6 Modalities as Lenses

Let s denote an underlying world state. Observing it through modality m yields

$$o^{(m)} = g_m(s) + \epsilon^{(m)}, \tag{8}$$

where g_m is a modality-specific measurement or projection and $\epsilon^{(m)}$ captures noise and information loss. An RGB image, speech waveform, text description, depth map, or motor signal is not the world itself; each is a partial lens on it.

Shared latent spaces are not new. Perceiver IO demonstrated a general latent-processing architecture for heterogeneous structured inputs and outputs [2]. The proposal here is more operational: the shared representation persists across interactions, evolves recurrently, and contains multiple processes with distinct update schedules.

Design principle. A modality is an interface to internal state, not necessarily the format of thought.

6.1 Video generation is multi-stream

Video generation makes the multi-stream requirement concrete. A model rendering two actors in conversation must preserve actor identity, per-actor pose and motion, turn-taking, voice-to-character binding, lip synchronization, camera motion, shared scene state, and ambient audio along one evolving timeline. Multi-person conversational video research exposes the binding problem directly: several audio streams must remain attached to the correct visible people over time [14]. Native audio-video

generators with synchronized dialogue provide a broader system-level example [15].

At the behavioral level, this capability demonstrates that one generator can coordinate several concurrently evolving processes. It does not establish that the implementation contains explicit or independently addressable latent stream slots; the internal representation may remain entangled. The architectural question is whether making those processes persistent, selective, and schedulable would improve control, efficiency, interruption handling, and consistency.

7 From Token Time to Event Time

At generation time, a standard autoregressive model couples each visible increment to a token position. Call this cadence *token time*. Null-token recurrence separates this clock from the number of model evaluations: several silent passes can advance compute while visible token time remains unchanged. A general agent may further maintain audio, vision, planning, memory, and motor processes with different update schedules:

$$t_{\text{token}} \neq t_{\text{compute}} \neq t_{\text{world}}. \tag{9}$$

Computation may be scheduled by events $e_1, e_2, \ldots$ that activate only relevant state components. A new acoustic observation may update speaker and semantic state; a sudden obstacle may activate vision, planning, and control; an internal null action may request another reasoning pass. If no external response is needed, no visible language token need be produced.

7.1 Observer-relative event order

Deterministic event processing still requires a reproducible order, but that order must not be confused with causation. Let $e_i \prec_c e_j$ mean that e_i can causally influence e_j. The relation is a partial order: some event pairs remain unrelated.

A declared observer O and reference frame $\mathcal{F}_O$ assign each modeled world event a coordinate time $t_{\mathcal{F}_O}(e)$. The event bus can then use the lexicographic key

$$K_O(e) = \big(t_{\mathcal{F}_O}(e),\, \tau_O(r_e),\, \iota(e)\big), \tag{10}$$

where r_e is the local receipt event, $\tau_O(r_e)$ is the observer's monotonic local clock at receipt, and $\iota(e)$ is a stable final tie-breaker. The induced order is

$$e_i \prec_O e_j \quad \Longleftrightarrow \quad K_O(e_i) <_{\text{lex}} K_O(e_j). \tag{11}$$

The frame and clock are admissible only when they preserve causal precedence:

$$e_i \prec_c e_j \implies e_i \prec_O e_j. \tag{12}$$

For causally unrelated events, $\prec_O$ records chronology in the selected frame without implying that one event caused the other. The result is a deterministic total extension of the causal partial order, conceptually related to the distinction between causal order and logical clock order in distributed systems [12].

The physics analogy requires one qualification. *Proper time* is measured by a clock along a single worldline. It can order the receipt events r_e observed locally by one agent, but it cannot by itself order remote events at different locations. That broader ordering requires coordinate time in a declared reference frame, including its convention for simultaneity [13].

Invariant precedence, declared chronology. Causal order is preserved when it exists. Otherwise the system records whose frame defines "earlier" and "later."

7.2 Causal masking is not a world clock

For a causal decoder, the autoregressive objective factorizes a sequence as

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}), \tag{13}$$

while the decoder applies an attention mask $M_{ij} = -\infty$ for $j > i$. Position i therefore cannot read subsequent token positions during that forward pass [16]. The mask is an externally supplied information-flow constraint over the chosen serialization, not a learned physical causal order among the events represented by the tokens.

This does not make the learned weights order-neutral. Training shapes them under the causal factorization, positional scheme, and data distribution, so they may encode order-sensitive regularities. The narrower point is that the mask itself is not a learned parameter; neither an explicit observer frame nor a world-event order follows from the weights alone.

By analogy with in-context conditioning, a frame specification $c_{\mathcal{F}}$ could select a readout without changing the weights [17, 18]:

$$y_{\mathcal{F}} = D_\theta(Z; c_{\mathcal{F}})\,. \tag{14}$$

If Z retains information compatible with several viewpoints, $c_{\mathcal{F}}$ may induce a frame-consistent serialization. This possibility is termed *frame-conditioned latent readout*. As an intuition rather than a mechanism claim, it resembles *holographic restoration*: the prompt acts like a boundary condition that reconstructs one view from distributed state. The metaphor does not imply that the weights literally implement a hologram, contain a complete frame-independent world, or preserve invariants without targeted training and evaluation.

8 Persistent Multi-Stream Latent State

At event e_n in the declared order, let the model maintain

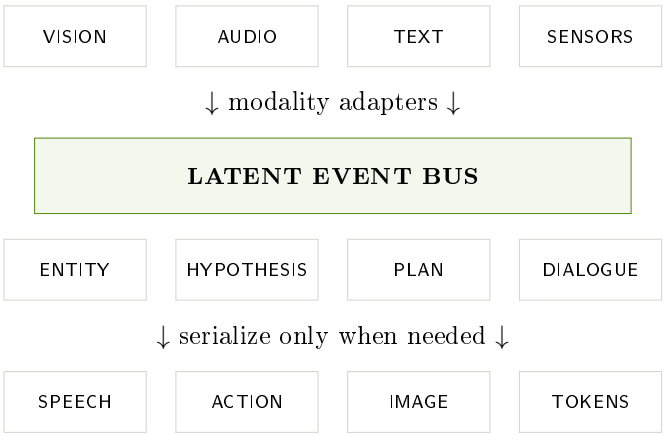
$$Z_n = \left[z_n^{(1)}, z_n^{(2)}, \ldots, z_n^{(K)} \right]. \tag{15}$$

Persistent streams may represent physical entities, goals, memory elements, candidate hypotheses, predicted futures, perceptual processes, or control processes. A routing policy produces an activity mask

$$a_n = \pi_{\text{route}}(Z_n, e_n), \qquad a_n \in \{0,1\}^K, \tag{16}$$

and updates only selected components:

$$z_{n+1}^{(k)} = \begin{cases} F_k\left(z_n^{(k)}, e_n, Z_n\right), & a_n^{(k)} = 1, \\ z_n^{(k)}, & a_n^{(k)} = 0. \end{cases} \tag{17}$$



The Latent Event Bus is a shared representational and routing layer that connects persistent latent streams to observations and outputs. Modality-specific encoders and decoders provide interfaces without prescribing the internal reasoning format. Null-token recurrence is one candidate mechanism for asking this state to spend additional compute without forcing an external serialization step.

9 Delayed Commitment

When several hypotheses remain plausible, they can remain components of one persistent state rather than becoming complete, independent textual trajectories. Recent work also scales latent reasoning through parallel stochastic trajectories and latent reward models [4]. Continuous representations do not uniquely enable alternatives; the narrower proposal is to maintain unresolved alternatives as coordinated components of a persistent world model.

The argument does not depend on a continuous vector containing unlimited information. Discrete output imposes a particular quantization and semantic commitment; delaying that commitment may preserve distinctions that matter to later decisions.

10 Relation to Prior Work

Coconut. Continuous hidden-state reasoning and latent alternatives are established [1]. The present proposal turns continuous carry into an action inside the existing decode loop and extends state across perception, reasoning, and action.

Perceiver IO. General latent processing across heterogeneous inputs and outputs is established [2]. The present proposal adds recurrent persistence and selective event-triggered updates.

Hybrid latent reasoning. Reinforcement learning can optimize latent and discrete reasoning policies [3]. Here it would train the null-step policy, latent carry, routing, and scheduling; RL itself is not the claimed novelty.

Parallel latent test-time scaling. Multiple latent trajectories can be sampled and ranked [4]. Here, by contrast, streams may represent entities, goals, modalities, or hypotheses within one state.

Multi-Stream LLMs. Several token streams can read and write in parallel [5]. Here internal streams need be neither tokenized nor synchronized to one update clock.

Looped / recurrent-depth Transformers. Reusing shared blocks can increase effective depth and enable reasoning gains or depth extrapolation [6, 7, 8]. The null-token proposal controls recurrence through a generation action and mixes a continuous side state into the next ordinary input path.

Intended novelty. The proposed combination is a null-controlled continuous recurrence primitive plus persistent structured latent multiplicity, event-driven scheduling, and explicit observer-relative event order.

11 Training

Stage I - conventional pretraining. Language or multimodal predictive training initializes useful representations and modality adapters.

Causal masking at this stage enforces left-to-right access within each serialized example; it does not supervise a unique physical chronology. Separating causal invariants from observer-dependent order therefore requires additional training signals, such as frame labels, causal graphs, or multi-frame renderings.

Stage II - explicit reasoning bootstrap. Existing supervised or reinforced CoT provides a stable reasoning policy. This stage is valuable precisely because it teaches variable-length computation through a readable discrete trace.

Stage III - continuous recurrence. Introduce the null action, latent carry projection, and gate. Distill from explicit CoT if desired, then optimize outcome quality

under a recurrence budget. The model learns when a visible token is useful and when another continuous step is worth its compute.

Stage IV - persistent multi-stream state. Add recurrent slots and event routing through next-state prediction, entity tracking, masked-state reconstruction, cross-modal prediction, or distillation. Outcome-driven optimization can then learn which components to update and when.

A generic objective can combine task success with penalties for unnecessary computation, unnecessary serialization, unstable recurrence, monitoring loss, and violations of declared event order:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_c C + \lambda_s S + \lambda_d D + \lambda_m M + \lambda_o O. \quad (18)$$

12 Experimental Program

The recurrence mechanism should first be tested independently of the larger architecture. A compute-matched study can compare (A) explicit text CoT, (B) fixed tied recurrent depth, (C) learned variable null-token recurrence, and (D) variable recurrence with persistent multi-stream state.

12.1 Null recurrence under matched compute

Hold total transformer FLOPs approximately constant. Compare accuracy, visible reasoning-token count, silent-step count, latency, and calibration. Sweep the maximum null-step budget and the penalty λ_n. If variable recurrence merely spends more compute without improving the frontier, it is not useful.

12.2 Gate and carry ablation

Compare token-only input, a null embedding without continuous carry, ungated carry, gated carry as in Eq. 4, and always-on gated carry as in Eq. 5. This separates the benefit of extra repeated depth from the benefit of carrying continuous state between passes.

12.3 Monitorability probe

Train auxiliary decoders or probes that reconstruct intermediate reasoning concepts, task subgoals, or predicted next visible CoT checkpoints from silent states. Measure decodability as the null budget increases. Test sparse explicit checkpoints as a compromise between full textual reasoning and completely latent recurrence.

12.4 Asynchronous multi-entity world

Construct an environment with independently evolving entities and heterogeneous update rates. As entity count increases, measure entity-state accuracy, retention, cross-stream interference, FLOPs, latency, and memory. Permute the delivery order of causally unrelated events while retaining frame timestamps; a correct implementation should reproduce the same observer-relative state trajectory.

12.5 Delayed evidence

Early observations support several hypotheses; later evidence resolves them. Measure hypothesis diversity, contradiction recovery, and revision cost.

12.6 Interruption

Begin an output or action and inject new information before completion. Measure reaction latency, reusable computation, consistency, and unnecessary recomputation. Compare whether silent latent steps can be reused more effectively than already-serialized CoT.

12.7 Cross-modal action without text

Provide vision and audio and require a non-linguistic action. Compare explicit text-mediated pipelines against direct latent-state-to-action behavior.

12.8 Multi-actor dialogue generation

Generate scenes with two or more actors whose lines overlap, alternate, or are interrupted. Measure identity persistence, speaker-to-voice binding, lip synchronization, turn-order accuracy, cross-actor leakage, and recovery from an unscripted event. Under matched compute, compare an entangled baseline with explicit actor-indexed persistent streams.

12.9 Frame and receipt-order ablation

Compare three schedulers: arrival order alone, chosen-frame coordinate time alone, and the causal-consistent key K_O. Measure replay stability, stale-state writes, causal inversions, and sensitivity to transport jitter. Repeat under alternative declared frames to distinguish invariant causal predictions from frame-dependent chronology.

12.10 Frame-conditioned reconstruction

Serialize the same causal event graph under two or more declared frames, then condition the model on $c_{\mathcal{F}}$ to reconstruct each view. Hold causal ancestry, entity identity, and outcomes fixed; vary only the coordinate order of unrelated events. Measure invariant accuracy, frame-dependent chronology, and round-trip consistency under $\mathcal{F}_1 \to \mathcal{F}_2 \to \mathcal{F}_1$.

Killer ablation. If variable null-token recurrence does not beat compute-matched explicit CoT and fixed-depth baselines, or if the continuous carry can be removed without loss, the recurrence primitive is not justified. If asynchronous structured state then fails to beat simpler sequential or parallel baselines on independent-clock and interruption tasks, the broader architecture is not justified either.

13 Failure Modes and Falsification

- **Overthinking:** excessive null recurrence degrades an otherwise correct state.
- **Null collapse:** the model emits too many or too few silent steps regardless of task difficulty.
- **Carry bypass:** the continuous side channel is ignored, so gains come only from extra repeated depth.
- **Latent drift:** recurrent updates push state outside the pretrained distribution.
- **Monitorability collapse:** useful reasoning moves into states that available probes cannot recover or audit.
- **Stream collapse:** multiple slots converge to effectively identical representations.
- **Fragmentation:** isolated streams fail to maintain global consistency.
- **Scheduling overhead:** routing costs exceed the savings from selective updates.
- **False concurrency:** parallel slots merely disguise a serial computation.
- **Frame leakage:** changing only the declared frame alters predictions that should remain invariant.
- **Prompted confabulation:** the model produces a plausible frame-specific narrative without a cross-frame-consistent latent event structure.
- **Order overclaim:** chronological order is misread as evidence of causation.

14 Discussion

Because text is abundant and next-token prediction scales well, language models encourage communication, reasoning, memory, and scheduling to be organized around one sequence. CoT made this constraint productive by turning token generation itself into a robust adaptive-compute loop. The same success now suggests a minimal relaxation: keep the loop, but stop requiring every recurrent state transition to pass through a semantic token.

The world does not arrive as a sentence. A room contains many entities at once. Perception need not pause during action, and planning need not stop when new evidence arrives. One hypothesis need not become a sentence before another can coexist with it. A silent null action can separate visible token time from compute time; persistent streams can further separate compute time from world time.

The price is that human readability can no longer be assumed from the existence of a reasoning loop. Once internal computation is allowed to proceed without serialization, monitorability has to be preserved deliberately rather than inherited accidentally from the token bottleneck.

Event time should not become another hidden global counter. The event bus instead requires a declared perspective: preserve causal precedence, identify the observer and reference frame that determine chronology, and distinguish local receipt time from modeled occurrence time.

Question for architecture. Why should language simultaneously serve as interface, reasoning trace, recurrent state carrier, memory representation, scheduler, and computational clock?

15 Conclusion

Chain-of-thought can be understood as a discrete recurrent loop whose variable length provides adaptive compute. Continuous latent reasoning shows that intermediate computation need not always be verbalized. Looped transformers show that tied recurrent depth can buy additional computation without additional stored parameters. The proposed null-token primitive combines these observations: when more computation is needed, emit an internal null action and carry a gated continuous state into the next ordinary transformer pass; when commitment is useful, return to a visible token.

This primitive fits naturally inside a Persistent Multi-Stream Latent World State: compute when an event requires computation, update the state that the event affects, preserve causal precedence, and serialize only when interaction requires it.

Tokens are interfaces. Null can be compute. Events move the world.

Revision History

v1.1 – September 2, 2026. Added null-token plus gated continuous recurrence; reframed CoT as discrete recurrence; separated fixed and variable recurrence; added continuous-RL and serving discussion; added monitorability analysis; added an explicitly speculative comparison with public reporting about Astra; expanded recurrent-depth prior work and falsification experiments.

v1.0 – August 21, 2026. Initial position paper covering persistent multi-stream latent world state, modalities as lenses, the Latent Event Bus, token/compute/world time, observer-relative event ordering, frame-conditioned readout, and compute-matched falsification tests. The last pre-v1.1 manuscript commit is 948318f.

References

[1] S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, and Y. Tian, "Training Large Language Models to Reason in a Continuous Latent Space," arXiv:2412.06769, 2024.

[2] A. Jaegle, S. Borgeaud, J.-B. Alayrac, et al., "Perceiver IO: A General Architecture for Structured Inputs and Outputs," ICLR, 2022.

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

[4] R. You, Y. Li, M. Liu, W. Wang, L. Nie, and W. Li, "Parallel Test-Time Scaling for Latent Reasoning Models," ACL, 2026.

[5] G. Su, Y. Yang, X. Li, and J. Geiping, "Multi-Stream LLMs: Unblocking Language Models with Parallel Streams of Thoughts, Inputs and Outputs," arXiv:2605.12460, 2026.

[6] H. Kohli, S. Parthasarathy, H. Sun, and Y. Yao, "Loop, Think, & Generalize: Implicit Reasoning in Recurrent-Depth Transformers," arXiv:2604.07822, 2026.

[7] S. Li, Y. Zhang, J. Guo, Q. Gu, and M. Wang, "DeepLoop: Depth Scaling for Looped Transformers," arXiv:2607.13491, 2026.

[8] Y. Fan, A. Svete, and K. Lee, "Bridging the Gap Between Latent and Explicit Reasoning with Looped Transformers," arXiv:2606.31779, 2026.

[9] S. Palazzolo, A. Efrati, and R. Drew, "OpenAI Technique in 'Astra' Model Sparks Security Concerns," The Information, September 1, 2026.

[10] OpenAI, "Path to Astra: Critical Capabilities and Frontier Safeguards," September 1, 2026.

[11] J. Pachocki, "I want to prevent a race into unmonitorability...," public post, September 2, 2026, https://x.com/merettm/status/2095023204993490967.

[12] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.

[13] A. Einstein, "On the Electrodynamics of Moving Bodies," Annalen der Physik, vol. 17, pp. 891–921, 1905.

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

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

[16] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," NeurIPS, 2017.

[17] T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.

[18] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An Explanation of In-Context Learning as Implicit Bayesian Inference," ICLR, 2022.