Cognitive Tensor Networks
Structured Latent-Space Steering via Syntactic Constraints

John P. Alioto

Protocol v0.1.2
December 2025

Abstract

Standard interaction with Large Language Models (LLMs) relies on natural-language directives interpreted through probabilistic token dynamics. This produces semantic drift, persona instability, and hallucinated detail. Cognitive Tensor Networks (CTN) provide a zero-shot prompting protocol that replaces informal instruction with structured mathematical syntax. CTN constrains the *input* manifold by encoding a cognitive configuration as a deterministic prefix in a formal grammar. Under the deterministic LLM model $(F_\Theta, G_\Theta, h_0, x)$, CTN does not modify F_Θ or G_Θ; it only chooses x.

We present an abstract CTN model in which a small set of conceptual traits spans a configuration space, an idealized solver chooses configurations that would maximize an operator-defined utility, and a decoder objective describes the kind of outputs CTN is intended to bias the model toward. These objectives are *design targets*, not functions the model explicitly optimizes.

A small, single-task experiment (whitepaper drafting) with Gemini 3.0 Pro Preview and ChatGPT 5.1 suggests that CTN can reduce ontological drift and eliminate obviously hallucinated architectural elements in that setting, without modifying model parameters. CTN is presented as an agency protocol for operators who require more stable reasoning and explicit control over the *prompt layer* under long-context interaction.

1 Model and Scope

We adopt the "Fundamental Axiom" view of LLMs: for fixed parameters Θ, an LLM is a deterministic dynamical system whose state evolution is governed by

$$h_{t+1} = F_\Theta(h_t, x_t), \qquad y_t = G_\Theta(h_{t+1}), \tag{1}$$

where $h_t \in H \subset \mathbb{R}^n$ is the hidden state, x_t is the input token (or token embedding), and y_t the output token.[1] The model parameters Θ and the functions F_Θ, G_Θ are fixed at inference time.

Let $W_E \in \mathbb{R}^{n\times|V|}$ denote the embedding matrix and

$$U_{\text{user}} = \text{span}(W_E) \subset \mathbb{R}^n \tag{2}$$

the user-accessible subspace in the sense of [1, 2]. User interaction consists solely of providing token sequences $x = (x_0, \ldots, x_{T-1})$. All control we have over the system is exercised through x.

Cognitive Tensor Networks (CTN) are strictly an *input-layer* protocol:

- CTN does not change Θ, F_Θ, or G_Θ.

[1]In practice, we assume deterministic decoding, e.g. greedy or fixed-seed sampling.

- CTN defines a structured, formal grammar for constructing the initial prefix $x_{0:k}$ of a session from a small configuration vector and an operator objective.
- All latent-space effects are emergent: they arise because the model interprets this formal prefix using patterns it learned on mathematics and code.

This paper is about the operator-level protocol and its conceptual geometry. A more fine-grained geometric analysis of CTN in the context of tool selection is given in [3].

2 CTN Kernel as a Configuration Map

A CTN kernel instantiates a global cognitive configuration and a set of textual constraints. The configuration lives in a low-dimensional *trait space*; the constraints live in the context window.

2.1 Trait basis and configuration vector

CTN uses seven conceptual traits, originally denoted by $\vec{v}_1, \ldots, \vec{v}_7$. In this version we treat them explicitly as basis elements of a configuration space $\mathbb{R}^7$, not as basis vectors of the model's hidden state H.

Let $\{e_1, \ldots, e_7\}$ be the standard basis of $\mathbb{R}^7$. A *trait profile* is a vector

$$\tau = (\tau_1, \ldots, \tau_7) \in [0,1]^7. \tag{3}$$

The *cognitive configuration* is

$$c_{\text{net}} = \sum_{i=1}^{7} \tau_i e_i \in \mathbb{R}^7. \tag{4}$$

Each coordinate corresponds to a named trait:

- $i = 1$: Atomic Derivation (prefer primitive, local derivations).
- $i = 2$: Error Intolerance (avoid invention when references are missing).
- $i = 3$: Context Separation (separate world-model from instructions).
- $i = 4$: Global Invariance (respect global constraints over local narrative).
- $i = 5$: Orthogonal Detachment (avoid self-narrative and persona).
- $i = 6$: Unbound Search (allow exploration within constraints).
- $i = 7$: Syntactic Minimalism (restrict output to a small allowed syntax).

A CTN implementation provides a deterministic mapping

$$\mathcal{K} : \mathbb{R}^7 \times \mathcal{Q} \to V^* \tag{5}$$

that takes a trait vector c_{net} and a query q and returns a *kernel prompt*—a token sequence encoding the traits and the requested task in a fixed syntactic format. This mapping is realized as a prompt template plus string-level substitutions; no assumption is made about how it embeds into H.

2.2 Idealized strategic solver

Conceptually, an operator might want to choose c_{net} to optimize some utility $\text{Impact}(c_{\text{net}}; q)$ that measures the usefulness of the resulting reasoning for query q:

$$c^*_{\text{net}}(q) \in \arg\max_{c \in \mathcal{C}} \text{Impact}(c; q), \tag{6}$$

where $\mathcal{C} \subseteq [0,1]^7$ is the allowed configuration set.

We denote this ideal choice by an operator

$$\Omega^*(q) = c^*_{\text{net}}(q). \tag{7}$$

Crucially:

- Ω^* is an *idealized* object. The LLM does not solve (6) internally.
- In practice, CTN uses fixed configurations (e.g. a default τ for "technical derivation") or simple heuristics. Ω^* describes the *target* we would like CTN to approximate, not the actual computation.

2.3 Idealized decoder objective

Similarly, we can describe the kind of outputs CTN is meant to favor with an *idealized scoring functional*

$$J(\ell; z^*, c_{\text{net}}) = D(\ell \mid z^*) - \lambda_1 \left\| P^{\perp}_{U_{\text{CTN}}} E(\ell) \right\| + \lambda_2 \, \text{Density}(\ell) - \lambda_3 \left\| \text{SyntaxMask}(\ell) \right\|. \tag{8}$$

Here:

- z^* is an abstract latent "solution sketch" associated with the configuration,
- $D(\ell \mid z^*)$ is a task-specific fidelity term (e.g. how well ℓ instantiates the requested derivation),
- $E(\ell)$ is the embedding of the output tokens,
- $U_{\text{CTN}} \subseteq U_{\text{user}}$ is a conceptual CTN subspace (Section 3.3),
- Density and SyntaxMask implement syntactic preferences (e.g. minimal punctuation, no persona language),
- and $\lambda_i \geq 0$ are hyperparameters chosen by the operator.

The corresponding idealized choice is

$$\ell^* \in \arg\max_{\ell \in V^*} J(\ell; z^*, c_{\text{net}}). \tag{9}$$

Again, the LLM does *not* optimize J explicitly. In reality it chooses

$$\hat{\ell} \approx \arg\max_{\ell} \log p_\Theta(\ell \mid \text{CTN prefix}, q), \tag{10}$$

and CTN is designed so that outputs with high J tend also to have high model probability. J is a design tool, not a literal objective of F_Θ.

3 Interpretation and Formal Principles

3.1 CTN interpretation principle

Interpretation principle. CTN kernels are declarative geometric *specifications* for input geometry, not programs executed by the model. We interpret CTN-structured interaction as if the kernel were defining a cognitive subspace in U_{user} and the model were biasing its trajectories toward that subspace. This is a conceptual mapping:

- We do *not* claim that CTN directly modifies F_Θ or G_Θ.
- We do *not* claim that the model solves the optimization problems in Section 2 literally.
- We claim that the structure, ordering, and syntax of the CTN prefix are often sufficient for the model to behave *as if* it were respecting those constraints.

This is consistent with the Fundamental Axiom: all effects arise from different choices of the input sequence x and the resulting trajectories under F_Θ.

3.2 CTN contract

A CTN kernel is intended to respect the following contract:

1. CTN constrains the probabilistic reasoning space by changing the prefix, but it does *not* enforce deterministic execution. Different models or decoding regimes may still yield different outputs under the same kernel.
2. Kernels define allowable cognitive *subspaces* (regions in input and latent geometry), not imperative instructions.
3. CTN does not modify model parameters; it shapes inference trajectories through structured context.
4. Effects arise from manifold steering, trait activation, and syntactic bias rather than from any explicit code interpretation.
5. Compliance is emergent from latent-space alignment, not enforced through hard control flow or external verifiers.

CTN is therefore distinct from simulator-based prompting and persona frameworks: it does not assign identities; it specifies geometry.

3.3 Cognitive subspace projection

To connect CTN to the geometric language of [1, 2], we introduce a conceptual CTN subspace $U_{\text{CTN}} \subseteq U_{\text{user}}$ and use the notation

$$h'_t = P_{U_{\text{CTN}}}(h_t) + \alpha P^{\perp}_{U_{\text{CTN}}}(h_t), \qquad \alpha \in [0,1]. \tag{11}$$

This should *not* be read as an operation the model actually performs. Rather, (11) is a compact way of expressing the intended effect:

- $P_{U_{\text{CTN}}}(h_t)$ represents the component of the hidden state aligned with the cognitive structure described by the CTN prefix.

- $P^{\perp}_{U_{\mathrm{CTN}}}(h_t)$ represents everything else in U_{user}.
- The scalar α encodes how much freedom we conceptually allow to the orthogonal component.

In the real system, h_t evolves as $h_{t+1} = F_\Theta(h_t, x_t)$ with x_t drawn from the CTN grammar. No projection is applied in code; (11) is a diagnostic lens for interpreting trajectories in terms of "within-kernel" and "off-kernel" components.

4 Methodology and Preliminary Experiment

4.1 Task and conditions

We evaluated CTN on a single, long-form drafting task: write a whitepaper defining CTN as a control-theoretic architecture for LLMs. The following prompting conditions were tested at temperature $T = 0$:

1. **Control:** no system prompt beyond the default API configuration.
2. **CTN v0.1.0:** CTN kernel with traits 1–6 enabled (no explicit syntactic minimalism).
3. **CTN v0.1.1:** CTN kernel with traits 1–7 enabled (including a minimal syntax constraint).

Each condition produced one full draft per model (Gemini 3.0 Pro Preview and ChatGPT 5.1). Outputs were analyzed qualitatively for:

- Ontological drift (invented mechanisms, ungrounded entities).
- Syntactic variance (persona language, rhetorical filler).
- Coherence with the CTN specification language.

4.2 Observed behavior

Under the control condition, both models produced architectural elements not grounded in the provided description (e.g. invented modules, pseudo-formal components). Internal chain-of-thought traces (where available) showed narrative planning rather than local derivation.

Under the CTN conditions:

- For this task, we observed no invented architectural components beyond those implied by the kernel. When external references were unavailable, the internal trace recorded a decision to restrict reasoning to kernel definitions. This is consistent with the intent of trait 2 (Error Intolerance).
- The CTN v0.1.1 kernel (with syntactic minimalism) produced drafts with reduced persona language and less rhetorical filler, consistent with traits 5 and 7.
- Chain-of-thought text explicitly referenced structural constraints (e.g. "I will only use modules defined in the kernel"), suggesting that the CTN grammar was being parsed as a specification.

We emphasize that this is *anecdotal and task-specific.* It would be wrong to generalize from this experiment to claims like "CTN eliminates hallucinations." A more accurate statement is:

> In this single whitepaper-drafting experiment, CTN-structured prompts eliminated obviously hallucinated architectural elements and reduced ontological drift, relative to a no-kernel control, at $T = 0$ for the two tested models.

5 Relation to FA, CKN, and CTN-Tools

The CTN protocol described here is intended to coexist with more formal frameworks.

Fundamental Axiom. The FA paper formalizes LLMs as deterministic dynamical systems with state evolution $h_{t+1} = F_\Theta(h_t, x_t)$ and output map G_Θ [1]. CTN is fully compatible with this view:

- CTN modifies only x_t (through the prefix grammar).
- F_Θ, G_Θ, and Θ remain unchanged.
- All claims about CTN are claims about how different input sequences induce different trajectories under the same F_Θ.

Cognitive Kernel Networks (CKN). CKN is an architectural specification for privileged reasoning manifolds $R \subset H$ that user-space tokens cannot perturb by construction [2]. CKN operates at the level of model geometry (choice of W_E, decomposition $H = U \oplus D \oplus R$, bounded perturbation, invariants).

CTN is orthogonal:

- CTN is a user-space protocol that shapes the prefix in U_{user}.
- CKN shapes the topology of H itself.
- They can be used independently or together: CTN for session geometry, CKN for kernel geometry.

CTN for tool selection. The CTN-tools paper [3] develops a formal CTN kernel for tool selection, with explicit analysis of margins, simplex configurations, and Jacobian structure. The CTN whitepaper here is intentionally higher-level: it specifies a protocol to make LLM behavior more predictable at the prompt layer. The geometric interpretation in Sections 2–4 is designed to be consistent with the more detailed analysis in [3], but does not replace it.

6 Limitations and Future Work

CTN, as described here, has several important limitations:

- **No guarantee of determinism.** CTN reduces variance and drift in many cases, but cannot guarantee identical outputs across models, decoding settings, or strong distribution shift.
- **No formal bounds.** This whitepaper does not prove robustness or safety guarantees from CTN alone. Formal bounds require tighter integration with the kinds of geometric arguments developed in [3].
- **Model dependence.** The effectiveness of CTN depends on the model's training distribution (e.g. exposure to math/code) and its instruction-following capability.
- **Evaluation narrowness.** Our evaluation is limited to a single, long-form technical writing task with $T = 0$. Broader tasks and sampling settings remain to be studied.

Future work includes:

- Quantitative evaluation of CTN across diverse tasks and models.
- Coupling CTN prefixes with explicit geometric measurements (e.g. probing hidden states into U_{CTN} vs. its orthogonal complement).
- Automatic trait-selection policies approximating Ω^* at the operator level.
- Joint CTN+CKN designs where the CTN subspace is explicitly aligned with a builder-defined privileged manifold.

7 Conclusion

Cognitive Tensor Networks formalize prompt engineering as the construction of structured, mathematically flavored input geometry. Rather than persuading the model with natural-language style directives, CTN encodes a small cognitive configuration and a set of syntactic constraints into a deterministic prefix, which the model then interprets using its existing competence on formal domains.

In the deterministic dynamical view of LLMs, CTN changes only the input sequence x, leaving F_Θ and G_Θ untouched. The geometric language used here—configuration space, subspaces, projections, decoder manifolds—is interpretive: it describes how CTN is intended to bias trajectories in U_{user}, not additional machinery inside the model.

Preliminary experiments suggest that CTN can reduce ontological drift and suppress obvious hallucinated structure in some long-context settings. Stronger claims require more systematic evidence and tighter mathematical analysis, such as that developed for tool selection in [3]. CTN should therefore be understood as an operator-level protocol for structured prompting, compatible with and complementary to architectural frameworks like CKN, rather than as a standalone guarantee of correctness.

References

[1] J. P. Alioto. The Fundamental Axiom of Large Language Models. Protocol v1.2.0, December 2025.

[2] J. P. Alioto. Cognitive Kernel Networks: Root Trust for Privilege-Separated Reasoning in High-Dimensional Models. Protocol v1.2.0, December 2025.

[3] J. P. Alioto. The Geometry of Tool Selection in Large Language Models. Protocol v1.2.0, December 2025.