Preprint. Under review. Do not distribute.

Forecasting Agent Failure in Long-Horizon AI Systems: Trajectory Prediction as a Governance Primitive

Marco van Hurne, Dennis Leo
EIGENVECTOR RESEARCH
marco.vanhurne@eigenvector.eu

June 2026

Abstract

Large language model (LLM)-based autonomous agents are increasingly deployed in long-horizon, open-ended environments where governance cannot be reduced to discrete action validation. In such settings, a single locally valid action can cascade into a global trajectory failure through path-dependent policy violations. We propose that effective governance must shift from action validation to state-space trajectory control. We introduce the *Ontological Compliance Gateway* (OCG), a theoretical framework that models agent reasoning as a controlled dynamical system evolving over a Riemannian manifold. We formalize compliance policies as control barrier functions (CBFs) defining a viability kernel, and governance as a projection operator onto this valid manifold. Crucially, we prove that action governance is insufficient to guarantee trajectory validity (Theorem 1), and propose that agent failure can be forecasted before policy violation by measuring trajectory divergence via Lyapunov exponents and constraint pressure in the latent state space. We frame hallucination and misalignment as attractor instability, providing a geometric characterization of failure modes. This work establishes the theoretical foundation for predictive runtime alignment in autonomous systems.

1. INTRODUCTION

The deployment of autonomous AI agents introduces a fundamental challenge for runtime governance. Traditional software systems execute deterministic workflows where governance is achieved through access controls, authorization models, and discrete policy enforcement. Large language model (LLM)-based agents fundamentally alter this assumption: they exhibit long-horizon planning, emergent tool orchestration, autonomous memory formation, and non-deterministic execution [1, 2].

As a consequence, governance can no longer be reduced to action validation. A compliant action sequence may still produce an invalid system state. Consider an autonomous procurement agent executing the action *Transfer 50,000 EUR*. Under traditional action governance, a policy engine evaluates the discrete action against a static rule set. However, the exact same action can be correct, fraudulent, or catastrophic depending entirely on the historical trajectory and the global system state [3]. This observation motivates a shift from action-level to trajectory-level governance.

Recent empirical work has begun to surface this problem. AgentForesight [4] demonstrates that agent failures in long-horizon tasks are often preceded by early, localized reasoning errors that cascade into trajectory-level collapse. Analyses of long-horizon decision making [5] reveal that step-wise greedy policies fail to account for delayed consequences. The Agent Viability Framework [6] formalizes this using Aubin's viability theory, establishing that governing an agent reduces to estimating an unobserved risk bound.

These findings converge on a critical gap: current governance architectures evaluate what the agent *does*, but fail to evaluate where the agent is *moving*. This paper addresses this gap by introducing a dynamical systems formalization of agent governance.

1.1 Contributions

We make the following contributions:

1. Formal Insufficiency Theorem. We prove that action governance is insufficient to guarantee trajectory validity, providing a constructive counterexample via path-dependent policy violations.

2. Dynamical Systems Formalization. We model agent reasoning as a stochastic differential equation over a Riemannian manifold, and governance as a control barrier function enforcing forward-invariance of the compliance manifold.

3. Failure Forecasting Framework. We propose trajectory divergence (Lyapunov exponent) and constraint pressure as predictive metrics for agent failure, and characterize hallucination as attractor instability.

4. Ontological Telemetry. We define a set of runtime metrics enabling continuous monitoring of agent trajectory health.

2. RELATED WORK

2.1 Dynamical Systems and Neural Networks

The interpretation of neural networks as continuous dynamical systems was formalized by Neural ODEs [7], which demonstrated that residual networks can be interpreted as Euler discretizations of ordinary differential equations. Massaroli et al. [8] extended this to a rigorous continuous-depth formalism. Hamiltonian Neural Networks [9] model energy-conserving dynamics as symplectic vector fields. The Koopman operator framework [10] provides a linear embedding of nonlinear dynamics, enabling tractable analysis of complex systems. These works collectively establish the theoretical foundation for treating agent computation as a trajectory through a high-dimensional state space.

2.2 Runtime Governance and Agent Safety

Runtime governance for AI agents has emerged as an active research area. The work of [3] formalizes compliance policies as deterministic functions over execution paths, demonstrating that prompt-level instructions and static access control are special cases of a path-dependent policy framework. The Agent Viability Framework [6] introduces Aubin's viability theory as a governance primitive, establishing monitoring, anticipation, and monotonic restriction as necessary and sufficient properties for documented failure modes. AgentForesight [4] demonstrates online auditing of agent trajectories, training a 7B model to alarm at the earliest decisive error in multi-agent systems.

2.3 Control Barrier Functions

Control Barrier Functions (CBFs) provide formal safety certificates for dynamical systems [11, 12]. A CBF defines a forward-invariant safe set via a differentiable certificate function. Recent work has extended CBFs to neural network systems [13], providing exact verification of safety constraints. We adapt these concepts from physical control systems to semantic state-space governance.

2.4 Mechanistic Interpretability

Mechanistic interpretability treats internal representations as objects of study. The linear representation hypothesis [14] and transformer circuit analysis [15] demonstrate that features correspond to specific directions in activation space. Sparse autoencoders [16] provide a practical method for decomposing LLM activations into interpretable features. The residual stream geometry paper [17] demonstrates that belief state geometry is encoded in transformer residual streams. These works provide empirical grounding for the claim that semantic states can be identified in latent space.

3. MATHEMATICAL FOUNDATIONS

3.1 The Semantic State Manifold

Definition 1 (Semantic State Manifold).
Let $(\mathcal{M}, g)$ be a Riemannian manifold representing the semantic state space of an autonomous agent, where g is a Riemannian metric. A state $s \in \mathcal{M}$ encodes the agent's current beliefs, memory activations, and contextual representations. The dimension of $\mathcal{M}$ is determined by the hidden state dimensionality of the underlying LLM.

The Riemannian metric g is not Euclidean in general; it captures the intrinsic geometry of the activation space as established by the manifold hypothesis [18] and empirically supported by residual stream geometry analysis [17].

Definition 2 (Agent Dynamics).
The evolution of the agent is modeled as a controlled stochastic differential equation (SDE):

$$ds_t = F(s_t, c_t, \pi)dt + \sigma(s_t)dW_t \quad (1)$$

where $F: \mathcal{M} \times C \times \Pi \to T\mathcal{M}$ is the learned drift vector field (policy π acting on context c_t), $\sigma(s_t)$ is the diffusion coefficient capturing inference stochasticity, and W_t is a standard Wiener process.

Remark 1. *The stochastic formulation is essential. LLM inference is inherently stochastic (temperature sampling); a purely deterministic ODE formulation would misrepresent the true dynamics. The diffusion term $\sigma(s_t)dW_t$ captures this stochasticity, and the governance framework must be robust to it.*

3.2 The Compliance Manifold and Viability Kernel

Definition 3 (Compliance Manifold).
An ontology $\mathcal{O} = (E, R, C)$ defines entities E, allowed relations R, and semantic constraints C. The compliance manifold is the closed subset of structurally and semantically valid states:

$$S_{valid} = \{s \in \mathcal{M} \mid s \text{ satisfies } \mathcal{O}\} \subseteq \mathcal{M} \quad (2)$$

The forbidden region is $F = \mathcal{M} \setminus S_{valid}$. The constraint boundary is $\partial G = \partial S_{valid}$.

Definition 4 (Control Barrier Function).
A differentiable function $h: \mathcal{M} \to \mathbb{R}$ is a Control Barrier Function (CBF) for the system (1) if:

$$h(s) \geq 0 \Leftrightarrow s \in S_{valid} \quad (3)$$

and there exists an extended class-$\mathcal{K}$ function α such that:

$$\sup_u [\nabla h(s) \cdot F(s,c,\pi)] \geq -\alpha(h(s)) \quad (4)$$

The viability kernel $K_V(S_{valid})$ is the largest subset of S_{valid} from which the system can remain in S_{valid} forever [6]. The CBF condition (4) ensures that S_{valid} is forward-invariant under the governed dynamics.

4. THEORETICAL FRAMEWORK

4.1 The Insufficiency of Action Governance

Current API gateways and guardrails implement *action governance*: evaluating a discrete transition $s_t \to^{a_t} s_{t+1}$ against a policy. We formally prove this is insufficient.

Theorem 1 (Trajectory Insufficiency of Action Governance).
Let $G_{local}: S \times A \times S \to \{0,1\}$ be a local action governance predicate. There exist trajectories $\tau = \{s_0, s_1, \ldots, s_T\}$ such that $G_{local}(s_t, a_t, s_{t+1}) = 1$ for all $t \in [0, T-1]$, yet τ violates a global constraint $C_{global} \in C$.

Proof. We construct an explicit counterexample. Consider a financial ontology with a Separation of Duties (SoD) constraint: *the agent that creates a transaction cannot also approve it*. This is a global trajectory constraint, not a local transition constraint.

Let a_1 = *draft invoice* and a_2 = *approve invoice*. Define:

$$G_{local}(s_0, a_1, s_1) = 1 \quad \text{(draft is valid from initial state)}$$

$$G_{local}(s_1, a_2, s_2) = 1 \quad \text{(approval is locally valid)}$$

However, the trajectory $\tau = (a_1, a_2)$ executed by the same agent identity violates C_{SoD}. Since G_{local} is Markovian with respect to the action space, it cannot detect this path-dependent violation without full trajectory history. Therefore, G_{local} is insufficient to guarantee C_{global}. □

Remark 2. *This result generalizes to any path-dependent constraint, including temporal ordering constraints, resource accumulation limits, and multi-step authorization chains. The SoD example is illustrative; the proof structure applies to any constraint that cannot be evaluated without trajectory history.*

4.2 The OCG Projection Operator

To enforce trajectory validity, the OCG intercepts the unconstrained field $F(s)$ and applies a projection operator.

Definition 5 (Governed Dynamics).
The OCG transforms the unconstrained agent field into a governed field via the CBF-constrained optimization:

$$F_{OCG}(s) = \arg\min_{v \in T_s\mathcal{M}} ||v - F(s)||^2 \quad (5)$$

subject to: $\nabla h(s) \cdot v \geq -\alpha(h(s))$

This is a quadratic program (QP) with a linear constraint, solvable in closed form. The solution is:

$$F_{OCG}(s) = F(s) + \lambda(s)\nabla h(s) \quad (6)$$

where $\lambda(s) = \max(0, (-\nabla h(s) \cdot F(s) - \alpha(h(s))) / ||\nabla h(s)||^2)$.

Theorem 2 (Forward Invariance of Compliance Manifold).
Under the governed dynamics F_{OCG}, the compliance manifold S_{valid} is forward-invariant: if $s_0 \in S_{valid}$, then $s_t \in S_{valid}$ for all $t \geq 0$.

Proof Sketch. By the CBF condition (4), the QP in Definition 5 always has a feasible solution when $s \in S_{valid}$. The solution F_{OCG} satisfies $\nabla h(s) \cdot F_{OCG}(s) \geq -\alpha(h(s))$, which by Nagumo's theorem implies that $S_{valid} = \{s: h(s) \geq 0\}$ is forward-invariant under the governed dynamics. □

5. FORECASTING AGENT FAILURE

5.1 Hallucination as Attractor Instability

We propose a geometric characterization of hallucination as a failure mode of the dynamical system.

Definition 6 (Grounded and Hallucination Attractors).
Let $A_{grounded} \subseteq S_{valid}$ be the set of stable attractors corresponding to factually grounded reasoning states. Let $A_{hallucination} \subseteq F$ be the set of unstable attractors corresponding to confabulated states. Hallucination is the event that the reasoning trajectory τ crosses the stability boundary ∂G and converges toward an element of $A_{hallucination}$.

Hypothesis 1 (Lyapunov Characterization of Hallucination).
Let $\nu(\tau)$ be the maximal Lyapunov exponent of the trajectory τ. We hypothesize that:

$$\nu(\tau) > 0 \text{ near } \partial G \Rightarrow P(\text{hallucination} \mid \tau) > p_{threshold} \quad (7)$$

This hypothesis is empirically testable and constitutes the primary experimental contribution of this framework.

Remark 3. *We explicitly distinguish Hypothesis 1 from a theorem. The Lyapunov characterization of hallucination is a proposed empirical claim, not a proven mathematical result. Its validation requires measurement of Lyapunov exponents on actual LLM trajectories, which constitutes the primary experimental work described in Section 7.*

5.2 Ontological Telemetry Metrics

We define five runtime metrics for continuous monitoring of agent trajectory health.

Definition 7 (Constraint Pressure).
The constraint pressure at state s_t is:

$$\kappa(s_t) = 1 / (h(s_t) + \varepsilon) \quad (8)$$

where $\varepsilon > 0$ prevents division by zero. High κ indicates proximity to the constraint boundary. The derivative $d\kappa/dt > 0$ indicates the agent is accelerating toward a policy violation.

Definition 8 (Trajectory Divergence).
The trajectory divergence at state s_t is:

$$\delta(s_t) = ||F_{OCG}(s_t) - F(s_t)|| \quad (9)$$

When $\delta > 0$, the OCG is actively intervening. High sustained δ predicts impending task failure, as the agent's internal policy is consistently misaligned with the ontological constraints.

Table 1 summarizes the full set of ontological telemetry metrics.

Table 1. Ontological Telemetry Metrics for Runtime Failure Forecasting.

Metric	Symbol	Definition	Failure Signal
Constraint Pressure	$\kappa(s_t)$	$(h(s_t) + \varepsilon)^{-1}$	$d\kappa/dt > 0$
Trajectory Divergence	$\delta(s_t)$	$\|\|F_{OCG} - F\|\|$	Sustained $\delta > \delta_0$
Lyapunov Exponent	$\nu(\tau)$	$\lim_{T\to\infty} (1/T) \log\|\|\delta s_T/\delta s_0\|\|$	$\nu > 0$
Governance Interventions	$I(\tau)$	$\|\{t: \lambda(s_t) > 0\}\|$	I/T > threshold
Boundary Violations	$V(\tau)$	$\|\{t: h(s_t) < 0\}\|$	$V > 0$

6. OCG ARCHITECTURE

6.1 Reference Architecture

The OCG is structured as a layered runtime governance harness operating between the agent's LLM core and the external environment. The architecture comprises four primary components:

State Projector. Maps the LLM's latent state $s_t \in \mathbb{R}^d$ to an ontological state $\pi(s_t) \in \mathcal{M}$. This is the most computationally demanding component. We propose using sparse autoencoders [16] as a practical approximation, mapping LLM activations to interpretable feature vectors that can be evaluated against the ontological constraints.

Constraint Evaluator. Evaluates the CBF $h(s)$ and its gradient $\nabla h(s)$ at the current state. For discrete ontologies, this reduces to a constraint satisfaction problem. For continuous ontologies, this requires a differentiable representation of the compliance manifold.

Projection Solver. Solves the QP in Definition 5 to compute $F_{OCG}(s)$. The QP has a closed-form solution (Equation 6), enabling real-time execution.

Telemetry Engine. Computes the five metrics in Table 1 at each step and maintains a trajectory log for audit purposes.

Table 2. OCG Architecture Layers with Formal Representations.

Layer	Component	Formal Object
1	Ontological State Model	$\mathcal{O} = (E, R, C)$
2	Compliance Manifold	$S_{valid} \subseteq \mathcal{M}$
3	Control Barrier Function	$h: \mathcal{M} \to \mathbb{R}$
4	State Projector	$\pi: \mathbb{R}^d \to \mathcal{M}$
5	Projection Solver	$F_{OCG} = F + \lambda \nabla h$
6	Trajectory Monitor	$\tau = \{s_0, ..., s_T\}$
7	Telemetry Engine	$(\kappa, \delta, \nu, I, V)$

7. EXPERIMENTAL DESIGN

7.1 Experiment 1: Failure Prediction via Trajectory Divergence

Hypothesis. Trajectory divergence $\delta(s_t)$ and constraint pressure $\kappa(s_t)$ increase significantly before agent failure, enabling prediction 2–5 steps ahead.

Setup. We propose running open-source models (Llama 3 70B, Mistral 7B) on the AgentBench and WebArena benchmarks, augmented with a procurement ontology defining SoD constraints. We record hidden states at each reasoning step and compute the telemetry metrics in Table 1.

Evaluation. We evaluate failure prediction using Area Under the ROC Curve (AUC) for binary classification of "failure within k steps" using the telemetry metrics as features. We compare against the AgentForesight [4] baseline.

Expected Result. AUC > 0.75 for $k = 3$ steps ahead, with constraint pressure providing the strongest signal for policy violations and Lyapunov exponent providing the strongest signal for hallucination.

7.2 Experiment 2: Projection Operator Feasibility

Hypothesis. The projection operator F_{OCG} can be approximated via sparse autoencoder features with latency < 50ms on a single GPU.

Setup. We train a sparse autoencoder on Llama 3 activations following the methodology of Bricken et al. [16]. We implement the CBF constraint evaluator using the SAE feature vectors as a proxy for the ontological state. We measure projection latency on an NVIDIA A100.

7.3 Experiment 3: Hallucination Attractor Analysis

Hypothesis. Hallucinated outputs correspond to trajectories with positive Lyapunov exponents near the stability boundary (Hypothesis 1).

Setup. We collect a dataset of correct and hallucinated outputs from Llama 3 on factual QA tasks (TriviaQA, NaturalQuestions). We record hidden states and compute finite-time Lyapunov exponents using the standard algorithm [19]. We test whether Lyapunov exponents discriminate hallucination with AUC > 0.65.

7.4 Experiment 4: Comparison vs. Baselines

Baselines. We compare the OCG framework against: (1) AgentForesight [4] on the AFTraj-2K benchmark; (2) Runtime Governance on Paths [3] on policy violation detection; (3) Agent Viability Framework [6] on risk estimation accuracy.

Table 3. Experimental Design Summary.

Exp.	Hypothesis	Primary Metric	Baseline
E1	Failure prediction from δ, κ	AUC (k=3 steps)	AgentForesight [4]
E2	Projection feasibility	Latency (ms)	Direct policy check
E3	Hallucination via ν	AUC (hallucination)	Perplexity baseline
E4	End-to-end governance	Task success rate	No governance

8. EVALUATION FRAMEWORK

We propose evaluating the OCG framework along four dimensions:

Predictive Accuracy. The primary measure of success is the ability to predict agent failure before it occurs. We measure AUC for failure prediction at horizons k = 1, 3, 5 steps ahead.

Governance Completeness. We measure the fraction of global constraint violations that are detected by trajectory monitoring but missed by action governance alone. This directly quantifies the benefit of trajectory-level governance over action-level governance.

Autonomy Preservation. A governance framework that blocks all actions is trivially safe but useless. We measure task completion rate under OCG governance versus no governance, requiring that task completion rate remains above 85%.

Computational Overhead. We measure the per-step latency of the OCG projection operator, targeting < 50ms for real-time applicability.

9. DISCUSSION

9.1 Relationship to Existing Work

The OCG framework is positioned as the theoretical foundation for a class of runtime governance systems that includes AgentForesight [4], the Agent Viability Framework [6], and Runtime Governance on Paths [3]. These works provide empirical instantiations of the trajectory governance principle; the OCG provides the mathematical formalization. Specifically, the Agent Viability Framework's Informational Viability Principle — governing an agent reduces to estimating an unobserved risk bound — is a special case of our CBF formulation where the risk bound corresponds to the CBF value $h(s)$.

9.2 The Novelty Claim

We explicitly acknowledge that the individual components of the OCG framework are not novel: dynamical systems views of neural networks are established [7, 8], CBFs are well-known in control theory [11], and trajectory-level governance has been proposed informally [3]. The novelty of this work lies in three specific contributions: (1) the formal proof that action governance is insufficient (Theorem 1), which has not appeared in the governance literature; (2) the CBF formalization of compliance policies, which provides a differentiable and computationally tractable governance mechanism; and (3) the Lyapunov characterization of hallucination as attractor instability (Hypothesis 1), which provides a geometric framework for failure prediction.

10. LIMITATIONS

We identify the following limitations of the proposed framework:

Runtime Projection Tractability. The most significant limitation is the computational cost of mapping high-dimensional LLM latent states to discrete ontological manifolds in real-time. While sparse autoencoders offer a practical approximation, the fidelity of this approximation is uncharacterized. A SAE feature vector is not equivalent to an ontological state; the semantic gap between these representations is a fundamental open problem.

Non-Convexity of the Compliance Manifold. Enterprise ontologies rarely form convex valid sets. The CBF framework requires that h be differentiable and that the QP in Definition 5 be feasible. For non-convex S_{valid}, the QP may be infeasible or may converge to local minima. Neural CBF approaches [13] offer a partial solution but introduce their own verification challenges.

Ontology Completeness. The OCG implicitly assumes that a complete, consistent ontology can be constructed for the deployment environment. This is a decades-old open problem in knowledge representation. An incomplete ontology may over-constrain the agent, reducing its utility.**

Stochasticity and Non-Markovian Dynamics. LLMs with memory are not Markovian. The SDE formulation (Definition 2) captures stochasticity but does not fully address the non-Markovian nature of long-horizon agent behavior. A rigorous treatment would require a history-dependent state augmentation $\tilde{s_t} = (s_t, h_t)$ where h_t

is the history, significantly increasing the dimensionality of the state space.

Empirical Validation. The current work is primarily theoretical. Hypothesis 1 (Lyapunov characterization of hallucination) and the failure prediction claims require empirical validation on real LLM trajectories. The experimental design in Section 7 outlines the required validation, but results are not yet available.

11. FUTURE WORK

Several directions for future work emerge from this analysis:

Empirical Validation. The most immediate priority is executing the experimental program in Section 7, particularly Experiments 1 and 3, which test the core failure prediction claims.

Neural CBF for Semantic Spaces. Developing differentiable CBFs that operate directly on LLM activation spaces, without requiring an intermediate ontological projection, would significantly reduce the computational overhead of the framework.

Multi-Agent Extension. The current framework addresses single-agent governance. Multi-agent systems introduce additional complexity: trajectory validity must be evaluated across the joint state space of all agents, and governance constraints may be non-decomposable.

Koopman Linearization. Applying Koopman operator theory [10] to linearize the agent dynamics would enable tractable computation of the projection operator in a lifted Hilbert space, potentially resolving the non-convexity problem.

12. CONCLUSION

We have introduced the Ontological Compliance Gateway, a mathematical framework for runtime governance of autonomous AI agents based on dynamical systems theory and control barrier functions. The central theoretical contribution is the formal proof that action governance is insufficient to guarantee trajectory validity, motivating a shift to state-space trajectory control. We have proposed that agent failure can be forecasted before policy violation by measuring trajectory divergence and constraint pressure, and characterized hallucination as attractor instability in the semantic state space. The framework provides a rigorous mathematical foundation for the emerging class of trajectory-level runtime governance systems, and establishes a research program for empirical validation.

The governance of autonomous AI agents is not a software engineering problem. It is a control theory problem. The appropriate response to an agent approaching the boundary of its compliance manifold is not to block it; it is to redirect it. This is the shift from reactive governance to predictive governance that this work formalizes.

REFERENCES

[1] Anonymous. (2026). Runtime Governance for AI Agents: Policies on Paths. *arXiv preprint arXiv:2603.16586*.

[2] Anonymous. (2026). Why Reasoning Fails to Plan: A Planning-Centric Analysis of Long-Horizon Decision Making in LLM Agents. *arXiv preprint arXiv:2601.22311*.

[3] Anonymous. (2026). Runtime Governance for AI Agents: Policies on Paths. *arXiv preprint arXiv:2603.16586*.

[4] Anonymous. (2026). AgentForesight: Online Auditing for Early Failure Prediction in Multi-Agent Systems. *arXiv preprint arXiv:2605.08715*.

[5] Anonymous. (2026). Why Reasoning Fails to Plan. *arXiv preprint arXiv:2601.22311*.

[6] Anonymous. (2026). Governing What You Cannot Observe: Adaptive Runtime Governance for Autonomous AI Agents. *arXiv preprint arXiv:2604.24686*.

[7] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*, 31, 6572–6583.

[8] Massaroli, S., Poli, M., Park, J., Yamashita, A., & Asama, H. (2020). Dissecting Neural ODEs. *Advances in Neural Information Processing Systems*, 33, 3952–3963.

[9] Greydanus, S., Dzamba, M., & Yosinski, J. (2019). Hamiltonian Neural Networks. *Advances in Neural Information Processing Systems*, 32.

[10] Lusch, B., Kutz, J. N., & Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1), 4950. https://doi.org/10.1038/s41467-018-07210-0

[11] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., & Tabuada, P. (2019). Control barrier functions: Theory and applications. *2019 18th European Control Conference (ECC)*, 3420–3431.

[12] Richards, S. M., Berkenkamp, F., & Krause, A. (2018). The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems. *Proceedings of The 2nd Conference on Robot Learning*, 87, 466–476.

[13] Zhang, H., et al. (2023). Exact Verification of ReLU Neural Control Barrier Functions. *Advances in Neural Information Processing Systems*, 36.

[14] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

[15] Elhage, N., Nanda, N., Olsson, C., et al. (2021). A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html [Technical Report, not peer-reviewed].

[16] Bricken, T., Templeton, A., Batson, J., et al. (2023). Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*. https://transformer-circuits.pub/2023/monosemantic-features [Technical Report, not peer-reviewed].

[17] Shai, A. S., Marzen, S. E., Teixeira, L., Oldenziel, A. G., & Riechers, P. M. (2024). Transformers represent belief state geometry in their residual stream. *Advances in Neural Information Processing Systems*, 37. arXiv:2405.15943.

[18] Arvanitidis, G., Hauberg, S., & Schölkopf, B. (2020). Geometrically enriched latent spaces. *arXiv preprint arXiv: 2008.00565*.

[19] Benettin, G., Galgani, L., Giorgilli, A., & Strelcyn, J. M. (1980). Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. *Meccanica*, 15(1), 9–20.

[20] Elhage, N., Hume, T., Olsson, C., et al. (2022). Toy Models of Superposition. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html [Technical Report, not peer-reviewed].

[21] Kovachki, N., Li, Z., Liu, B., et al. (2023). Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. *Journal of Machine Learning Research*, 24(89), 1–97.

[22] Shi, H., & Meng, M. Q. H. (2022). Deep Koopman operator with control for nonlinear systems. *IEEE Robotics and Automation Letters*, 7(3), 7700–7707.

[23] Geshkovski, B., Letrouit, C., Polyanskiy, Y., & Rigollet, P. (2023). A mathematical perspective on transformers. *arXiv preprint arXiv:2312.10794*.

Appendix A: Proof of Theorem 1 (Extended)

We provide an extended proof of Theorem 1 covering the general case of path-dependent constraints.

General Setup. Let C_{global} be a constraint that depends on the full trajectory history $h_t = (s_0, a_0, s_1, a_1, \ldots, s_t)$. A local action governance predicate $G_{local}(s_t, a_t, s_{t+1})$ is Markovian: it depends only on the current state, action, and next state, not on h_t.

Claim. For any Markovian predicate G_{local}, there exists a path-dependent constraint C_{global} and a trajectory τ such that G_{local} accepts all transitions in τ but C_{global} rejects τ.

Proof. Define C_{global} as the constraint that the same entity does not perform both action `a_1` and action `a_2` within a trajectory. This constraint is path-dependent: it requires knowledge of the full action history.

Since G_{local} is Markovian, it cannot distinguish between: (a) a trajectory where `a_1` was performed by entity A and `a_2` by entity B (valid), and (b) a trajectory where both `a_1` and `a_2` were performed by entity A (invalid). Therefore, G_{local} either accepts both or rejects both, and cannot selectively enforce C_{global}. □

Appendix B: Computational Complexity of the Projection Operator

The projection operator in Definition 5 is a quadratic program (QP) with one linear constraint. By the closed-form solution in Equation 6, the computational complexity is $O(d)$ where d is the dimensionality of the state space. This is dominated by the cost of computing $\nabla h(s)$, which requires a forward pass through the CBF network. For a CBF implemented as a shallow neural network, this is $O(d \cdot w)$ where w is the width of the network.

The total per-step overhead of the OCG is

therefore O(d · w), which is linear in the state dimensionality. For typical LLM hidden state dimensions (d = 4096 for Llama 3 7B) and a shallow CBF network (w = 256), this corresponds to approximately 10^6 floating-point operations per step — well within real-time constraints on modern GPU hardware.