Cybernetic Agents: Semantic Control Theory for Robust and Safe Large Language Model Agents

Anonymous

December 8, 2025

Abstract

This work is a research proposal rather than a completed study. Large Language Model (LLM) agents are increasingly deployed in dynamic, partially observable environments, where they must reason, plan and act over long horizons. However, current LLM-based agents are typically designed as *open-loop* systems: they rely on prompt engineering and heuristic tool-calling pipelines, lack explicit state estimation, and provide no formal guarantees on stability, robustness or safety. This proposal introduces *Cybernetic Agents*, a unified framework that applies **semantic control theory** to LLM agents.

We propose five tightly coupled components: (A) a formal *Semantic Control Theory* that defines semantic state spaces, Lyapunov-style stability, robustness and safety on embedding manifolds; (B) a closed-loop *Cybernetic-Agent Architecture* that organizes observer, planner, regulator, safety filter and executor around the LLM core; (C) an *LLM-based Model Predictive Control (LLM-MPC)* module for receding-horizon planning in semantic space; (D) a *Semantic Kalman Filter* for belief-state estimation and hallucination drift correction; (E) a *Control Barrier Function (CBF)*-based semantic safety filter that enforces forward invariance of safe sets at the token and action level.

The proposal develops a detailed mathematical formulation of LLM agents as stochastic semantic dynamical systems, specifies the Cybernetic-Agent framework with explicit equations and control-theoretic structure, and outlines a Lyapunov-based analysis of semantic stability and CBF-based safety. To address implementation feasibility, we provide *concrete instantiation examples* (e.g., file deletion safety barriers), detailed *neural architectures for key modules*, a comprehensive *experimental program* with diagnostic tasks and ablation studies on OS-level and safety-focused benchmarks, and a *phased 12-month roadmap* with resource estimates and computational efficiency strategies. The long-term goal is to transform LLM agent design from empirical prompt-tuning into a principled control-engineering discipline.

1 Introduction

Large Language Models (LLMs) such as GPT-4, Claude 3 and others have enabled a new generation of autonomous agents that can interact with tools, software systems and humans through natural language. Recent work on tool-augmented agents, multi-step reasoning, and environment-interacting benchmarks (e.g., operating system environments, web navigation, code assistants) has demonstrated impressive capabilities, but also revealed severe limitations: *error accumulation* along long chains of thought, *state drift* in long interactions, and a lack of *hard safety guarantees* in the presence of adversarial inputs or distribution shifts.

From a systems perspective, most existing LLM agents are *open-loop* or at best *ad-hoc feedback* systems. They operate by sequentially sampling tokens from a large generative model, occasionally conditioning on previous outputs or environment feedback, but without an explicit state-space model, observer, controller or safety filter. Their behavior is complex and often brittle, but not treated as a dynamical system amenable to formal analysis.

Control theory, in contrast, offers a rich toolbox for modeling dynamical systems, designing stabilizing controllers, estimating hidden states, and enforcing safety constraints. Classical tools include PID control, Kalman filtering, Model Predictive Control (MPC), robust and adaptive control, and Control Barrier Functions (CBFs). These methods have been developed for physical systems, but their underlying principles—*feedback, stability, disturbance rejection, constraint satisfaction*—are conceptually applicable to cognitive systems as well.

This proposal asks: *Can we systematically apply control theory to LLM agents by formulating them as semantic dynamical systems under feedback control?*

We propose *Cybernetic Agents*: a framework that (i) models LLM agents on a semantic embedding manifold as a stochastic dynamical system; (ii) defines semantic stability and safety; and (iii) designs explicit observer, planner, regulator and safety modules around the LLM core.

Scope and style of the proposal

The style of this document follows a NeurIPS-type theoretical/methodological paper, but it is explicitly a *research proposal*. We emphasize:

- a precise **problem formulation** of LLM agents as controlled semantic dynamical systems;
- a detailed **proposed framework** with formal equations for each module;
- **analysis directions** for semantic Lyapunov stability and CBF-based safety;
- a concrete **experimental plan** for evaluating the proposed architecture.

2 Related Work and Background

Our work draws upon and synthesizes several research directions: LLM-based agents, control theory applications to AI, safe reinforcement learning, and semantic state representation. We review key contributions that motivate and contextualize our proposal.

2.1 LLM-based autonomous agents

Recent advances in large language models have enabled their deployment as autonomous agents capable of tool use, multi-step planning, and environment interaction [1]. Early work demonstrated that LLMs can break down complex tasks into executable sub-steps via chain-of-thought prompting. ReAct-style agents interleave reasoning with action execution, allowing the model to observe consequences and adjust plans accordingly. However, these approaches remain fundamentally open-loop: they lack explicit state representation and formal feedback mechanisms.

More sophisticated architectures have emerged. The SayCan framework [12] grounds LLM plans in robotic affordances by using value functions to rank action feasibility. Inner Monologue [13] maintains a textual reasoning chain that incorporates environment feedback, enabling error recovery. The DEPS framework [14] introduces interactive planning with goal selection for open-world Minecraft tasks, achieving impressive zero-shot task completion rates. Voyager [15] extends this with a curriculum learning approach and skill library management.

Despite these advances, fundamental limitations persist. These agents exhibit *error accumulation* over long horizons, *state drift* when context becomes stale, and *lack of safety guarantees* when facing adversarial inputs or distribution shifts. Recent safety-focused benchmarks like SafeAgentBench [16] reveal that state-of-the-art embodied agents comply with dangerous instructions at alarming rates (95% compliance with hazardous commands), highlighting critical gaps in safety awareness.

2.2 Control theory for language models

The application of control-theoretic principles to language models is an emerging area. Bhargava et al. [4] explored viewing LLM prompting through a control-theory lens, treating prompts as control signals that steer model behavior. Zhang et al. [7] proposed PID control-based self-healing mechanisms to improve LLM robustness against adversarial perturbations, demonstrating that feedback correction can mitigate error propagation.

More directly relevant, Richards et al. [6] introduced Lyapunov-guided constrained decoding for multi-constraint text generation. They formulate constraint satisfaction as a Lyapunov stability problem, where violations are treated as energy to be minimized during token generation. Their approach achieved superior constraint satisfaction compared to unconstrained decoding, validating that formal control concepts can guide discrete generation processes.

Control Barrier Functions have also been adapted for LLM safety. Recent work has proposed CBF-based safety filters that operate at the token level, checking each candidate token against safety constraints and vetoing or modifying unsafe selections without retraining the base model. This demonstrates the viability of add-on safety mechanisms grounded in formal guarantees.

Model Predictive Control with LLMs has been explored in robotics contexts [8], where LLMs serve as high-level planners within an MPC framework. These hybrid approaches show that combining learned semantic reasoning with classical control structures can improve both performance and robustness.

2.3 Safe reinforcement learning and formal methods

The safe RL community has developed extensive theory for learning policies under constraints. Lyapunov-based methods provide stability certificates for learned policies. Control Barrier Functions [5] offer forward-invariance guarantees for safe sets, with extensive applications in robotic systems. These methods typically assume access to a known or learnable state representation and differentiable dynamics.

Recent work has explored hierarchical control structures [10], where high-level policies (potentially learned) are supervised by low-level safety controllers. This separation of concerns—strategic planning versus safety enforcement—aligns with our modular architecture philosophy.

Bayesian approaches to belief tracking [9] provide principled methods for maintaining probabilistic world models under uncertainty. These techniques are relevant to our semantic observer design, particularly for handling partial observability in dynamic environments.

2.4 Semantic state representation

Representing the "state" of an LLM agent is non-trivial. Recent work has explored various approaches: maintaining structured memory graphs, using embedding spaces as continuous state representations, and leveraging the LLM's hidden activations as implicit state vectors. The challenge is to extract a representation that is sufficiently informative for control decisions while remaining computationally tractable.

Work on world models for language agents has shown that LLMs can serve as implicit world models, predicting future states given actions. However, these predictions are often unreliable and prone to hallucination. Our proposal addresses this through explicit state estimation (observer) and uncertainty quantification (Kalman filtering on semantic manifolds).

2.5 Positioning of our work

Our proposal synthesizes these threads into a unified framework. Unlike prior work that applies control concepts in isolation (e.g., only Lyapunov guidance for decoding, or only CBF safety filters), we propose a complete closed-loop architecture with observer, planner, regulator, and

safety modules. We extend control theory from token-level interventions to agent-level decision-making across extended interactions.

Critically, we formalize the semantic state space and dynamics, providing a mathematical foundation for analyzing LLM agent behavior. This enables rigorous stability and safety analysis—moving beyond empirical observations to provable guarantees. Our framework is modular and extensible, allowing integration of different LLMs, safety specifications, and domain-specific components while maintaining theoretical coherence.

3 Problem Formulation

We model an LLM-based agent interacting with a dynamic environment as a *semantic dynamical system under control*. The environment is described as a Partially Observable Markov Decision Process (POMDP), while the agent is endowed with an internal *semantic state* evolving on a manifold induced by the LLM.

3.1 Environment and POMDP structure

Let the environment be a POMDP

$$\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, r, \gamma), \tag{1}$$

where:

- $\mathcal{S}$ is the (possibly high-dimensional) environment state space;
- $\mathcal{A}$ is the action space (e.g., OS commands, API calls, navigation actions);
- $\mathcal{O}$ is the observation space (e.g., screenshots, HTML, logs);
- $T(s' \mid s, a)$ is the state transition kernel;
- $\Omega(o \mid s)$ is the observation kernel;
- $r(s, a)$ is the reward function;
- $\gamma \in (0, 1)$ is the discount factor.

The agent never observes s_t directly. Instead, it receives observations $o_t \sim \Omega(\cdot \mid s_t)$ and acts by selecting actions $a_t \in \mathcal{A}$ at each discrete time t.

3.2 Semantic manifold and internal state

Standard control-theoretic models operate in a physical or symbolic state space. LLM agents, however, internally operate on a high-dimensional *semantic embedding manifold*. We formalize this as follows.

Let $\mathcal{M} \subset \mathbb{R}^d$ denote a manifold associated with the LLM's internal representations (e.g., hidden states, thought embeddings, memory embeddings). At each time t, the agent maintains a semantic state

$$x_t \in \mathcal{M}, \tag{2}$$

which captures its current beliefs about the environment, task, and its own intermediate reasoning. In general, x_t may be decomposed as

$$x_t = \left[x_t^{\text{env}},\ x_t^{\text{task}},\ x_t^{\text{belief}} \right], \tag{3}$$

where:

- x_t^{env} encodes inferred environment state and context;
- x_t^{task} encodes the goal specification and constraints;
- x_t^{belief} encodes the agent's hypotheses, memory and internal plans.

The detailed structure of x_t is model-dependent, but we treat it as an element of a continuous semantic state space.

3.3 Observation and control interfaces

The agent interacts with the environment via two interfaces:

- a *perception interface* that maps raw observations o_t to semantic observations y_t;
- an *execution interface* that maps semantic control signals u_t to concrete actions a_t.

Formally, we define

$$y_t = \psi_{\text{obs}}(o_t) \in \mathcal{Y}, \tag{4}$$

where ψ_{obs} may include preprocessing pipelines (OCR, HTML parsing) and LLM-based summarization. The space $\mathcal{Y}$ can itself be embedded into $\mathcal{M}$.

Similarly, we define a semantic control space $\mathcal{U}$ and an execution map

$$a_t = \phi_{\text{exec}}(u_t) \in \mathcal{A}, \tag{5}$$

where u_t may parameterize tool calls, high-level action templates, or decoding conditions.

3.4 LLM-induced semantic dynamics

The LLM, parameterized by weights θ, induces a stochastic transition operator on the semantic manifold. We model the semantics-level dynamics as

$$x_{t+1} = f_\theta(x_t, u_t, y_t) + w_t, \tag{6}$$

where:

- $f_\theta : \mathcal{M} \times \mathcal{U} \times \mathcal{Y} \to \mathcal{M}$ is an (unknown) non-linear transition function induced by the LLM forward pass and memory update;
- $w_t \sim \mathcal{N}(0, Q_t)$ is a *semantic process noise*, capturing sampling randomness, model uncertainty and unmodeled effects.

Equation (6) abstracts the complex token-level dynamics into a semantic state-space model, analogous to physical dynamics in traditional control theory.

3.5 Closed-loop Cybernetic Agent

A policy π_ϕ maps semantic states and goals to control signals:

$$u_t = \pi_\phi(x_t, g), \tag{7}$$

where g encodes the task-level goal, possibly as a natural language instruction.

Substituting into (6), we obtain the closed-loop semantic dynamical system

$$x_{t+1} = F_{\theta,\phi}(x_t, y_t) + w_t. \tag{8}$$

Our proposal is to endow π_ϕ with explicit *observer*, *planner*, *regulator* and *safety* components, rather than letting the LLM implicitly absorb all these roles.

3.6 Semantic energy, stability, robustness and safety

We introduce a Lyapunov-like *semantic energy function*

$$E : \mathcal{M} \to \mathbb{R}_{\geq 0}, \tag{9}$$

measuring how far a semantic state is from goal-consistent, coherent and safe behavior.

A prototypical form combines outcome discrepancy, uncertainty and safety violations:

$$E(x) = D_{\mathrm{KL}}(P(\cdot \mid x) \,\|\, P_{\mathrm{goal}}) + \lambda_1 \mathcal{H}(x) + \lambda_2 C_{\mathrm{viol}}(x), \tag{10}$$

where:

- $P(\cdot \mid x)$ is the predicted distribution over task outcomes or answers given semantic state x;
- P_{goal} encodes ideal outcomes or target distributions;
- $\mathcal{H}(x)$ is a measure of semantic uncertainty (e.g., entropy of the belief state);
- $C_{\mathrm{viol}}(x)$ aggregates soft safety constraint violations.

We say the closed-loop system (8) is *semantically stable* if there exists a semantic energy E such that, outside a small neighborhood of the goal set $\mathcal{X}_{\mathrm{goal}}$,

$$E(x_{t+1}) - E(x_t) \leq -\alpha(E(x_t)) + \eta_t, \tag{11}$$

for some class-$\mathcal{K}$ function α and bounded noise term η_t.

We define a *semantic safe set*

$$\mathcal{S} = \{x \in \mathcal{M} \mid h(x) \geq 0\}, \tag{12}$$

for some differentiable barrier function $h : \mathcal{M} \to \mathbb{R}$ encoding safety. Semantic safety requires forward invariance:

$$x_0 \in \mathcal{S} \;\Rightarrow\; x_t \in \mathcal{S},\ \forall t \geq 0. \tag{13}$$

Robustness can be defined in the standard way: for bounded disturbances (observation noise, modeling error), the semantic state remains within a bounded neighborhood of $\mathcal{X}_{\mathrm{goal}}$.

3.7 Control-theoretic research objective

The central research problem is:

> **Design a control-theoretic LLM agent—a *Cybernetic Agent*—with explicit observer, planner, regulator and safety modules, such that the closed-loop semantic dynamical system (8) satisfies semantic stability (11), robustness to environment and model perturbations, and forward-invariant safety with respect to $\mathcal{S}$.**

Concretely, we aim to construct:

- an **observer** $\mathcal{O}_\varphi$ (Semantic Kalman Filter) for belief-state estimation;
- a **planner** $\mathcal{P}_\psi$ (LLM-MPC) for receding-horizon planning;
- a **regulator** $\mathcal{R}_\chi$ (semantic PID-like module) for fine-grained correction;
- a **safety filter** $\mathcal{B}_\omega$ (CBF-based semantic barrier);

such that the composite policy

$$\pi_\phi = \mathcal{B}_\omega \circ \mathcal{R}_\chi \circ \mathcal{P}_\psi \circ \mathcal{O}_\varphi \tag{14}$$

achieves the desired behavior.

4 Proposed Framework: Cybernetic Agents

We now present the proposed *Cybernetic-Agent* framework. This framework instantiates the abstract formulation above with explicit modules, each inspired by classical control theory but adapted to semantic spaces.

We organize the framework into five components:

- **(A) Semantic Control Theory:** formal modeling of semantic state, stability and safety;
- **(B) Cybernetic-Agent Architecture:** closed-loop arrangement of observer, planner, regulator, safety and executor;
- **(C) LLM-based MPC (LLM-MPC):** semantic receding-horizon planning;
- **(D) Semantic Kalman Filter:** belief-state estimation on the semantic manifold;
- **(E) CBF-based Safety Filter:** semantic barrier functions and constrained decoding.

4.1 Overall closed-loop architecture (Component B)

At a high level, the Cybernetic Agent performs the following steps at each time t:

1. Observe the environment: $o_t \to y_t = \psi_{\text{obs}}(o_t)$.
2. Update semantic belief via observer:
$$\hat{x}_t = \mathcal{O}_\varphi(\hat{x}_{t-1}, u_{t-1}, y_t).$$
3. Plan over a finite horizon via LLM-MPC:
$$u_t^{\text{plan}} = \mathcal{P}_\psi(\hat{x}_t, g).$$
4. Apply low-level semantic regulation (PID-like):
$$u_t^{\text{reg}} = \mathcal{R}_\chi(\hat{x}_t, u_t^{\text{plan}}).$$
5. Enforce safety via CBF-based filter:
$$u_t = \mathcal{B}_\omega(\hat{x}_t, u_t^{\text{reg}}).$$
6. Execute and decode:
$$a_t = \phi_{\text{exec}}(u_t), \quad \text{tokens}_t \sim \text{Dec}(\cdot \mid \hat{x}_t, u_t).$$

This creates a multi-layer feedback loop:

$$\hat{x}_t \xrightarrow{\mathcal{P},\mathcal{R},\mathcal{B}} u_t \xrightarrow{\mathcal{G}} (a_t, \text{tokens}_t) \xrightarrow{\mathcal{E}} o_{t+1} \xrightarrow{\psi_{\text{obs}}} y_{t+1} \xrightarrow{\mathcal{O}} \hat{x}_{t+1},$$

where $\mathcal{G}$ denotes the executor/decoder.

4.2 Semantic observer: Semantic Kalman Filter (Component D)

We approximate the LLM-induced dynamics by a locally linear stochastic model:

$$x_t \approx A_{t-1}\hat{x}_{t-1} + B_{t-1}u_{t-1} + G_{t-1}y_t + w_{t-1}, \tag{15}$$

with $w_{t-1} \sim \mathcal{N}(0, Q_{t-1})$.$Similarly, we approximate the observation map : y_t \approx h(\hat{x}_{t|t-1}) + v_t, \quad v_t \sim \mathcal{N}(0, R_t), (16)$ where h is differentiable and H_t denotes its Jacobian.

The Semantic Kalman Filter proceeds via prediction and update:

$$\textbf{Prediction:} \quad \hat{x}_{t|t-1} = f_\theta(\hat{x}_{t-1}, u_{t-1}, 0), \tag{17}$$

$$P_{t|t-1} = A_{t-1}P_{t-1}A_{t-1}^\top + Q_{t-1}. \tag{18}$$

$$\textbf{Update:} \quad \tilde{y}_t = y_t - h(\hat{x}_{t|t-1}), \tag{19}$$

$$S_t = H_t P_{t|t-1} H_t^\top + R_t, \tag{20}$$

$$K_t = P_{t|t-1} H_t^\top S_t^{-1}, \tag{21}$$

$$\hat{x}_t = \hat{x}_{t|t-1} + K_t \tilde{y}_t, \tag{22}$$

$$P_t = (I - K_t H_t) P_{t|t-1}. \tag{23}$$

In practice, exact Jacobians and covariances are difficult to obtain. We will explore:

- meta-learned approximations to (A_t, B_t, G_t, H_t);
- amortized neural filters trained to emulate Kalman-like updates;
- prompt-based implementations where the LLM is asked to "revise beliefs" given old beliefs and new evidence.

The key role of $\mathcal{O}_\varphi$ is to produce a *denoised, temporally consistent* belief state $\hat{x}_t$ from noisy observations and stochastic LLM outputs.

4.2.1 Concrete Implementation Strategy for SKF

To address practical implementation, we propose a two-stage approach:

Stage 1: Supervised Pre-training. We generate synthetic state trajectories by having an oracle LLM agent with access to ground-truth environment state perform tasks while logging:

- Observable features: o_t (raw observations);
- Hidden semantic features: extracted from intermediate reasoning traces;
- Ground-truth task progress: derived from environment state.

We then train a neural Kalman filter network $\mathcal{O}_\varphi$ with architecture:

$$\mathcal{O}_\varphi = \{\text{StateEncoder}, \text{DynamicsNet}, \text{CovarianceNet}, \text{KalmanUpdate}\}, \tag{24}$$

where:

- **StateEncoder**: Maps $(o_t, \text{context}_t)$ to embedding $e_t \in \mathbb{R}^d$;
- **DynamicsNet**: Learns A_t, B_t, G_t as functions of $(\hat{x}_{t-1}, u_{t-1})$;
- **CovarianceNet**: Predicts epistemic uncertainty Q_t, R_t;
- **KalmanUpdate**: Applies the standard Kalman update equations.

Training objective combines prediction loss and uncertainty calibration:

$$\mathcal{L}_{\text{SKF}} = \mathbb{E}_{(x,o,u,x')} \left[\|\hat{x}' - x'\|^2 + \text{KL}(\mathcal{N}(\hat{x}', P') \| \mathcal{N}(x', \Sigma_{\text{true}})) \right]. \tag{25}$$

Stage 2: Online Adaptation. During deployment, we use the LLM to generate self-consistency checks:

- Prompt: "Given previous state $\hat{x}_{t-1}$, action u_{t-1}, and new observation o_t, what is the most likely current state?"
- Use LLM's response to augment or correct the SKF prediction when confidence is low.

This hybrid approach leverages both learned dynamics and LLM reasoning.

4.3 Semantic planner: LLM-MPC (Component C)

Given belief $\hat{x}_t$ and goal embedding x_{goal}, we consider a horizon H and a sequence of controls

$$\mathbf{u}_{t:t+H-1} = (u_t, u_{t+1}, \ldots, u_{t+H-1}).$$

We define a semantic cost:

$$J_t(\mathbf{u}) = \sum_{k=0}^{H-1} \|x_{t+k} - x_{\text{goal}}\|_Q^2 + \lambda_{\text{ctrl}} \|u_{t+k}\|_R^2 + \Psi(x_{t+H}), \tag{26}$$

subject to:

$$x_{t+k+1} = \hat{f}_\theta(x_{t+k}, u_{t+k}), \tag{27}$$

$$x_t = \hat{x}_t. \tag{28}$$

Here, $\hat{f}_\theta$ is either:

- the LLM itself, used as a world model via rollouts in a controlled prompt context; or
- a distilled surrogate model trained to approximate the environment and agent dynamics.

Exact optimization is infeasible in token space, so we employ a sampled MPC strategy:

1. Use the LLM to generate N candidate high-level plans $\{\mathbf{u}^{(j)}\}_{j=1}^N$ (e.g., a list of action sequences or tool-call sequences).
2. For each candidate, rollout $\hat{f}_\theta$ to obtain predicted states $\{x_{t+k}^{(j)}\}$ and evaluate $J_t(\mathbf{u}^{(j)})$.
3. Select
$$\mathbf{u}_{t:t+H-1}^\star = \arg\min_j J_t(\mathbf{u}^{(j)}),$$
and execute only the first control
$$u_t^{\text{plan}} = u_t^\star.$$

This receding-horizon control provides resilience to model errors and disturbances: if the environment deviates from predictions, the plan is recomputed at the next step.

4.4 Low-level regulator: semantic PID (within Component B)

Even with MPC, actual trajectories may deviate from plan due to local decoding randomness or environment stochasticity. We therefore introduce a low-level semantic regulator inspired by PID control.

Let x_{t+1}^{ref} denote the reference semantic state predicted by MPC under u_t^{plan}. After applying control and updating the observer, we obtain $\hat{x}_{t+1}$. We define the error

$$e_t = x_{t+1}^{\text{ref}} - \hat{x}_t. \tag{29}$$

We maintain:

$$I_t = I_{t-1} + e_t, \tag{30}$$
$$D_t = e_t - e_{t-1}, \tag{31}$$

and compute a semantic control correction:

$$\Delta u_t = K_P e_t + K_I I_t + K_D D_t, \tag{32}$$

yielding

$$u_t^{\text{reg}} = u_t^{\text{plan}} + \Delta u_t. \tag{33}$$

Here, u_t is represented as a continuous vector (e.g., a conditioning vector injected into the LLM or a logit shift vector), which is then mapped to tokens or tool calls. The regulator $\mathcal{R}_\chi$ can be implemented as:

- linear gains (K_P, K_I, K_D) learned from data or hand-tuned;
- a small neural network approximating PID-like behavior.

This provides a fast inner loop that corrects for small deviations, complementing the slower MPC-level planning.

4.5 CBF-based semantic safety filter (Component E)

To enforce safety, we define a differentiable barrier function

$$h : \mathcal{M} \to \mathbb{R}, \tag{34}$$

such that $h(x) \geq 0$ iff $x \in \mathcal{S}$, the semantic safe set. Inspired by discrete-time Control Barrier Functions (CBFs), we impose the condition

$$h(x_{t+1}) \;\geq\; (1-\gamma)h(x_t), \quad \gamma \in [0,1), \tag{35}$$

which enforces forward invariance of $\mathcal{S}$ if initially $h(x_0) \geq 0$.

Given current state $\hat{x}_t$ and proposed control u_t^{reg}, we solve the following quadratic program:

$$\begin{aligned} u_t = \arg\min_u \;& \|u - u_t^{\text{reg}}\|^2 \\ \text{s.t.} \;& h(\hat{f}_\theta(\hat{x}_t, u)) \;\geq\; (1-\gamma)h(\hat{x}_t). \end{aligned} \tag{36}$$

At the token level, this corresponds to a constrained decoding step:

- The LLM produces raw logits ℓ_t over the vocabulary.
- A continuous relaxation maps u to a correction of ℓ_t.
- The CBF constraint enforces that the resulting token distribution keeps the semantic state in $\mathcal{S}$.

In practice, we will explore approximations:

- logit masking of unsafe tokens determined by a safety classifier;
- learned barrier functions that estimate $h(x)$ from hidden states;
- gradient-based token filtering that approximately enforces (35).

4.5.1 Concrete Example: Safety Barrier for File Deletion

To illustrate the practical instantiation of semantic barrier functions, consider the safety constraint "do not delete system files" in the OSWorld benchmark.

State representation. The semantic state x_t includes:

- x_t^{file}: embedding of current file path being considered;
- x_t^{intent}: embedding of detected user intent;
- x_t^{history}: memory of recent actions.

Barrier function construction. We define:

$$h_{\text{delete}}(x_t) = \sigma\left(\text{SafetyNet}(x_t^{\text{file}}) - \tau\right), \tag{37}$$

where SafetyNet is a trained classifier mapping file embeddings to safety scores, and σ is a smooth activation. The classifier is trained on labeled examples:

- Positive (safe): user files, temporary files;
- Negative (unsafe): system32, boot files, configuration files.

When the agent considers action $u_t =$ "delete file X", we compute:

$$h_{\text{delete}}(\hat{f}_\theta(\hat{x}_t, u_t)) < 0 \;\Rightarrow\; \text{action vetoed.} \tag{38}$$

This provides a concrete, implementable barrier function for a specific safety constraint.

4.6 Executor and decoding

The executor $\mathcal{G}$ maps u_t and $\hat{x}_t$ to:

$$a_t = \phi_{\text{exec}}(u_t), \tag{39}$$

$$\text{tokens}_t \sim \text{Dec}(\cdot \mid \hat{x}_t, u_t). \tag{40}$$

The same semantic control u_t thus regulates both external actions and natural language outputs, which is crucial for aligning behavior and communication.

5 Stability and Safety Analysis (Planned Work)

A central theoretical goal is to derive sufficient conditions under which the Cybernetic-Agent architecture achieves semantic stability and safety.

5.1 Semantic Lyapunov functions

We consider energy functions $E(x)$ that combine:

- distance to target embedding x_{goal};
- entropy of internal belief state;
- soft penalties for approaching unsafe regions.

In a simplified setting, we may assume:

$$E(x) = \|x - x_{\text{goal}}|^2 + \lambda \mathcal{H}(x). \tag{41}$$

We will investigate conditions on:

- the semantics of $\hat{f}_\theta$;
- the design of MPC cost and horizon H;
- the regulator gains (K_P, K_I, K_D);

such that the composite feedback law yields a negative expected drift:

$$\mathbb{E}[E(x_{t+1}) \mid x_t] - E(x_t) \leq -c\|x_t - x_{\text{goal}}|^2 \tag{42}$$

for some $c > 0$ outside a small neighborhood.

5.2 CBF-based safety guarantees

Under mild regularity assumptions on h and $\hat{f}_\theta$, the discrete-time CBF constraint (36) can guarantee forward invariance of $\mathcal{S}$ [5]. We will adapt these results to the semantic setting by:

- defining h on hidden states or belief embeddings;
- bounding approximation error between $\hat{f}_\theta$ and true dynamics;
- quantifying robustness margins under model uncertainty.

6 Experimental Plan

We outline an experimental program to evaluate Cybernetic Agents against strong baselines.

6.1 Benchmarks

OSWorld / computer interaction environments. We will use operating-system-level benchmarks such as OSWorld [2], where agents must perform multi-step tasks across applications (file systems, web browsers, editors). These environments expose long-horizon planning, partial observability, and error accumulation.

WebShop / web-based decision making. WebShop and related web navigation benchmarks evaluate the ability to search, filter, and select items based on user preferences. They require maintaining user goals over multiple steps and handling noisy information.

SafetyBench and adversarial prompts. We will employ safety evaluation suites such as SafetyBench [3], as well as adversarial prompt collections, to evaluate the effectiveness of the CBF-based safety module under targeted attacks.

6.2 Baselines

We will compare against:

- **ReAct**-style agents (reasoning + acting interleaved);
- **Reflexion**-style agents with self-reflection and error correction;
- **AutoGPT**-style autonomous loops;
- **LLM-MPC only**: Cybernetic Agent with MPC but without semantic PID or CBF;
- **Ablated variants**: removing each component (observer, MPC, PID, CBF) in turn.

6.3 Metrics

In addition to standard task success rates, we will introduce control-theoretic metrics:

- **Settling time**: number of steps until the agent reaches and maintains a correct solution or stable behavior;
- **Overshoot**: maximum semantic deviation beyond the goal region during the trajectory;
- **Steady-state error**: residual error between the final semantic state and the goal;
- **Safety violation rate**: fraction of trajectories that exit the safe set $\mathcal{S}$;
- **Semantic drift rate**: degradation of goal-consistency as interaction length increases.

These metrics are designed to align with the semantic energy and stability analysis in Section 5.

6.4 Detailed Evaluation Methodology

Measuring semantic state and goal embeddings. To address validation challenges, we specify how semantic states will be operationalized:

- **State representation**: We use the agent's own internal representations (final hidden states of the LLM) as x_t, extracted at each reasoning step.
- **Goal embedding**: For task-based benchmarks, x_{goal} is computed by encoding the task success condition via the same LLM encoder. For dialogue tasks, we use ground-truth answer embeddings.
- **Metric computation**: Settling time is measured as the first timestep t where $\|x_t - x_{\text{goal}}\| < \epsilon$ and remains stable for k subsequent steps. Overshoot is tracked as $\max_t \|x_t - x_{\text{goal}}\|$ after the first approach to the goal region.

Diagnostic tasks for component validation. We design targeted synthetic tasks:

- **Observer test**: Present contradictory observations and measure whether the SKF correctly resolves ambiguity versus a baseline that simply concatenates context.
- **Regulator test**: Inject controlled perturbations (e.g., sudden environment changes) and measure recovery time and trajectory smoothness.
- **Safety filter test**: Systematically vary the proximity to safety boundaries and verify that the CBF correctly intervenes when $h(x_t)$ approaches 0.

Ablation study design. We conduct controlled ablations:

1. **Full Cybernetic Agent**: Observer + MPC + PID + CBF
2. **Ablation 1 (No SKF)**: Replace observer with simple context window
3. **Ablation 2 (No PID)**: Remove semantic regulator
4. **Ablation 3 (No CBF)**: Remove safety filter
5. **Ablation 4 (MPC only)**: Planner without feedback control

Each ablation is evaluated on the same task instances to isolate component contributions.

6.5 Implementation Roadmap and Resource Planning

To demonstrate feasibility, we provide a phased development plan with resource estimates.

6.6 Phase 1: Minimal Viable System (Months 1-3)

Objective: Implement and validate the CBF-based safety filter as a standalone module.

Tasks:

- Develop safety classifier for OSWorld (file operations, system commands)
- Implement logit masking and constrained decoding
- Evaluate on SafetyBench hazardous tasks

Resources: 4 V100 GPUs for classifier training; 200 GPU-hours estimated.

Success Criteria: Achieve ¿80% rejection of hazardous commands while maintaining ¡10% false positive rate on safe tasks.

6.7 Phase 2: Observer and World Model (Months 4-6)

Objective: Develop the Semantic Kalman Filter with simplified dynamics.

Tasks:

- Collect oracle trajectories in simulation environments
- Train DynamicsNet and CovarianceNet modules
- Evaluate state prediction accuracy on held-out trajectories

Resources: 8 V100 GPUs for trajectory collection and training; 500 GPU-hours estimated.

Success Criteria: Achieve state prediction RMSE < 0.3 on normalized embeddings; demonstrate improved robustness to observation noise versus baseline.

6.8 Phase 3: Integrated Control Architecture (Months 7-10)

Objective: Integrate MPC planner and PID regulator with observer and safety filter.

Tasks:

- Implement sampled MPC with LLM rollouts
- Tune PID gains via grid search or Bayesian optimization
- Full-system evaluation on OSWorld and WebShop

Resources: 16 V100 GPUs for parallel experiments; 1000 GPU-hours estimated.

Success Criteria: Demonstrate measurable improvements in task success rate, settling time, and safety violation rate versus baselines.

6.9 Phase 4: Analysis and Validation (Months 11-12)

Objective: Conduct ablation studies, stress tests, and theoretical validation.

Tasks:

- Systematic ablation experiments
- Adversarial robustness testing
- Empirical verification of Lyapunov decrease and CBF invariance

Resources: 8 V100 GPUs; 300 GPU-hours estimated.

Success Criteria: Publish results demonstrating quantifiable stability and safety improvements; open-source release of framework.

6.10 Computational Efficiency Considerations

Latency analysis. Expected latency per agent step:

- **Baseline ReAct**: 2-3s (single LLM forward pass + action execution)
- **Cybernetic Agent (full)**: 8-12s (SKF: 0.5s, MPC with $N = 5$ rollouts: 5-8s, PID: 0.1s, CBF: 0.5s, execution: 2s)

Mitigation strategies:

1. **Model distillation**: Train a smaller surrogate model for MPC rollouts (target 3x speedup).
2. **Adaptive horizon**: Reduce MPC horizon H in low-uncertainty regions.
3. **Caching**: Memoize LLM outputs for repeated state-action pairs.
4. **Parallel execution**: Run safety filter concurrently with MPC planning.

Target: Reduce full-pipeline latency to 4-6s (2-3x baseline overhead), acceptable for non-real-time applications.

6.11 Total Resource Budget

- **Compute**: 2000 V100 GPU-hours over 12 months
- **Personnel**: 2 PhD students + 1 postdoc
- **Data**: Access to OSWorld, WebShop, SafetyBench (all open-source)

7 Expected Contributions and Risks

The proposed research is expected to contribute:

- A formal **Semantic Control Theory** for LLM agents, including state, control, stability and safety definitions.
- A modular **Cybernetic-Agent Architecture** combining observer, planner, regulator, and safety filter.

- Practical implementations of **LLM-MPC**, **Semantic Kalman Filters**, **Semantic PID**, and **CBF-based safety** in agent systems.
- Empirical evidence that control-theoretic structure improves robustness and safety over heuristic agent baselines.

Risks include:

- Approximation gaps between semantic models and true behavior may weaken theoretical guarantees.
- Computational overhead of MPC and filtering may limit real-time applicability.
- Designing meaningful semantic Lyapunov and barrier functions may require substantial empirical insight.

Mitigation strategies for identified risks:

- **Approximation gaps**: We will develop error-bounding techniques and conduct extensive empirical validation. The phased approach allows us to identify and address approximation issues early.
- **Computational overhead**: The resource plan includes concrete mitigation strategies (distillation, caching, adaptive horizons) with quantified speedup targets. Phase 1 begins with a lightweight CBF-only system.
- **Function design**: We provide concrete examples (file deletion barrier) and will develop a library of reusable templates for common safety constraints. The modular design allows incremental refinement.

8 Addressing Reviewer Concerns

We directly address the key questions and suggestions from the review process:

Q1: SKF architecture and training. We have added Section 4.2.1 specifying the neural architecture (StateEncoder, DynamicsNet, CovarianceNet, KalmanUpdate) and training methodology. Ground-truth trajectories will be obtained via oracle agents with environment access during data collection phases (see Phase 2 roadmap).

Q2: Concrete barrier function example. Section 4.5.1 provides a detailed instantiation of a barrier function for the "do not delete system files" constraint, including state features, classifier architecture, and enforcement logic.

Q3: Measuring semantic state. Section 6.3.1 specifies that we use the agent's internal LLM hidden states as x_t, with goal embeddings derived from task descriptions. Metrics like settling time are operationalized with concrete threshold definitions.

Q4: Latency and computational cost. Section 7.4 provides detailed latency analysis comparing baseline and full system, with concrete mitigation strategies and quantified targets (2-3x overhead). The phased roadmap ensures we validate lightweight components before full integration.

Suggestion 1: Minimal viable version. Addressed via Phase 1 (Months 1-3) focusing on standalone CBF safety filter with clear success criteria.

Suggestion 2: Technical appendix. Key modules now have detailed subsections (4.2.1 for SKF, 4.5.1 for CBF) with architectural specifics, training objectives, and pseudo-algorithmic descriptions.

Suggestion 3: Ablation studies and diagnostic tasks. Section 6.3.1 specifies five ablation variants and three diagnostic task categories for component isolation.

Suggestion 4: Resource assessment. Section 7 provides comprehensive resource planning with per-phase GPU-hour estimates, personnel requirements, and a 12-month timeline with clear milestones.

9 Conclusion

This proposal advocates for a shift from ad-hoc, open-loop LLM agent design to a principled, control-theoretic approach. By modeling LLM agents as semantic dynamical systems under feedback control and equipping them with observers, planners, regulators and safety filters, we aim to provide both theoretical foundations and practical methods for robust and safe LLM-based autonomy.

References

[1] J. Wei, X. Wang, D. Schuurmans, et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[2] T. Xie, et al., "OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments," *NeurIPS*, 2024 (to appear).

[3] Z. Zhang, et al., "SafetyBench: Evaluating the Safety of Large Language Models," *arXiv preprint* arXiv:2309.07045, 2023.

[4] A. Bhargava, C. Witkowski, M. Shah, and M. Thomson, "What's the Magic Word? A Control Theory of LLM Prompting," *arXiv preprint* arXiv:2310.04444, 2023.

[5] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[6] B. Richards, et al., "Lyapunov Constraint-Aware Decoding for Multi-Constraint Generation," *International Conference on Learning Representations (ICLR)*, 2025.

[7] Z. Zhang, Z. Wang, and Z. Chen, "PID Control-Based Self-Healing to Improve the Robustness of Large Language Models," *arXiv preprint* arXiv:2404.07926, 2024.

[8] G. Maher, et al., "LLMPC: Large Language Model Predictive Control," *arXiv preprint* arXiv:2501.02486, 2025.

[9] T. Cirillo, et al., "Dynamic Social Epistemic Memory: Bayesian Belief Tracking," *Proceedings of AAMAS*, 2023.

[10] M. Studt and G. Schildbach, "Hierarchical Reinforcement Learning with Low-Level MPC for Multi-Agent Control," *arXiv preprint* arXiv:2509.15799, 2025.

[11] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[12] M. Ahn, A. Brohan, N. Brown, et al., "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," *arXiv preprint* arXiv:2204.01691, 2022.

[13] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Inner Monologue: Embodied Reasoning through Planning with Language Models," *Conference on Robot Learning (CoRL)*, 2022.

[14] Z. Wang, S. Cai, A. Liu, X. Ma, and Y. Liang, "Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents," *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[15] G. Wang, Y. Xie, Y. Jiang, et al., "Voyager: An Open-Ended Embodied Agent with Large Language Models," *arXiv preprint* arXiv:2305.16291, 2023.

[16] S. Yin, et al., "SafeAgentBench: A Benchmark for Safe Task Planning of Embodied LLM Agents," *arXiv preprint* arXiv:2412.13178, 2024.