Universal Semantic Systems: A Meaning-First Architecture for Deterministic AI Systems

Robert Hansen
Universal Semantic Systems

Abstract

Large language models generate fluent output but lack a semantic substrate capable of maintaining meaning, consistency, or alignment across extended tasks. Existing work in tool-use, agent frameworks, and prompting heuristics attempts to compensate by layering procedural logic on top of probabilistic text generation. However, these approaches do not resolve the core issue: LLMs operate without a stable representational layer.

This paper introduces Universal Semantic Systems (USS) — an architecture that sits above foundation models and provides deterministic structure for meaning formation, planning, governance, and execution. USS consists of: (1) UST, a universal semantic token model; (2) USR, a runtime coordinating reasoning, routing, and posture; (3) SCP, a semantic control protocol for directive routing; and (4) ORCH-C, a deterministic orchestrator for multi-step agentic behavior. Together, these layers convert LLMs from probabilistic text engines into auditable semantic systems.

We outline the theoretical basis, describe the runtime architecture, compare USS to prior work, and present an example execution path. The result is a unified framework for meaning-first AI systems capable of long-horizon reliability, transparency, and alignment.

1 Introduction

Foundation models have transformed the landscape of AI, but their core design produces inherent limitations for complex or long-horizon reasoning tasks. They operate as probabilistic text engines without an underlying semantic substrate, meaning they lack continuity of meaning, stable internal representation, and deterministic governance.

Scaling laws work has shown that model capability increases predictably with compute, data, and parameter count [5], but scaling alone does not introduce semantic structure. Tool-use and agent frameworks extend LLM behavior, yet they do so on top of a probabilistic base that was never designed to maintain explicit meaning.

Universal Semantic Systems (USS) fills this gap. Rather than viewing the LLM as the system, USS treats the LLM as a linguistic executor inside a much larger semantic-computational architecture. USS introduces:

- UST — a universal token model defining semantic types.
- USR — a runtime that governs routing, meaning continuity, posture, and planning context.
- SCP — a control protocol for structured intent and directive flow.
- ORCH-C — a deterministic planner enforcing structure, invariants, and multi-step agentic reasoning.

Together these layers enable AI systems that reason deterministically, maintain semantic invariants, and uphold explicit governance rules.

2 USS Overview

Figure 1 provides a high-level overview of USS components and how they relate to the foundation model.

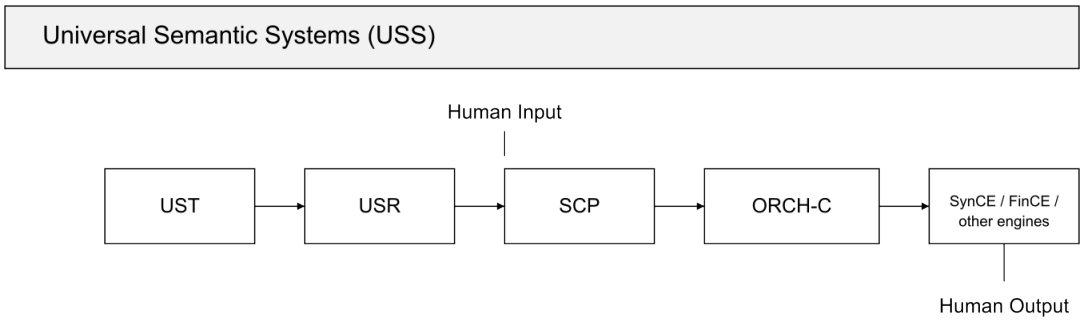

Figure 1: High-level USS architecture.

USS replaces prompt engineering with a structured semantic control layer. It unifies semantics, planning, posture, and governance into a single runtime design.

3 UST: Universal Semantic Token Model

UST defines the typed semantic substrate used across USS. Tokens represent intentions, constraints, postures, actions, entity types, and governance requirements. Unlike lexical tokens, UST tokens are meaning-first.

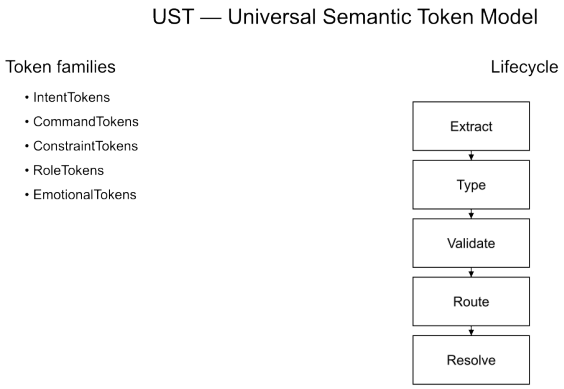

Figure 2: UST semantic token model.

UST tokens serve as the primary interface for all downstream modules including the runtime (USR), the control protocol (SCP), and the planner (ORCH-C).

4 USR: Universal Semantic Runtime

USR provides the execution environment for meaning-driven computation. It performs posture selection, intent routing, invariant enforcement, and planning context formation.

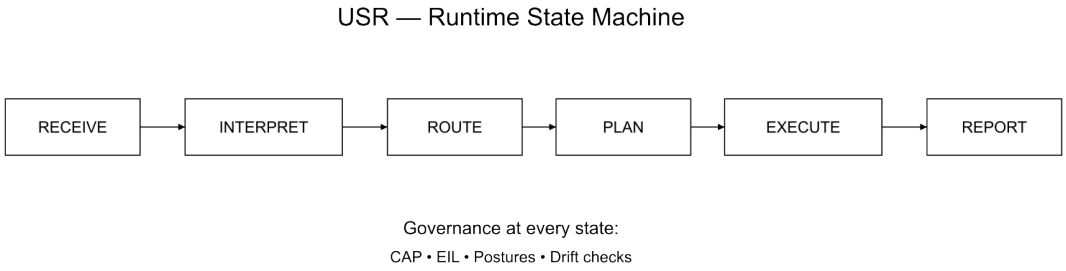

Figure 3: USR layer: the universal semantic runtime.

USR is responsible for maintaining semantic continuity across steps, guaranteeing that meaning does not drift as execution proceeds.

5 SCP: Semantic Control Protocol

SCP governs the flow of directives, transforming semantic tokens into structured operations routed through the runtime.

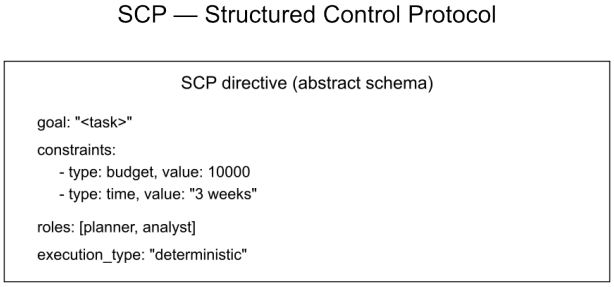

Figure 4: SCP directive routing model.

SCP ensures that all requests are typed, validated, and semantically grounded before reaching ORCH-C or the model executor.

6 ORCH-C: Deterministic Planner

ORCH-C converts semantic inputs into multi-step plans. It enforces structure, invariants, posture, reasoning rules, and governance.

ORCH-C provides determinism, traceability, and auditability for multi-step reasoning and agentic workflows.

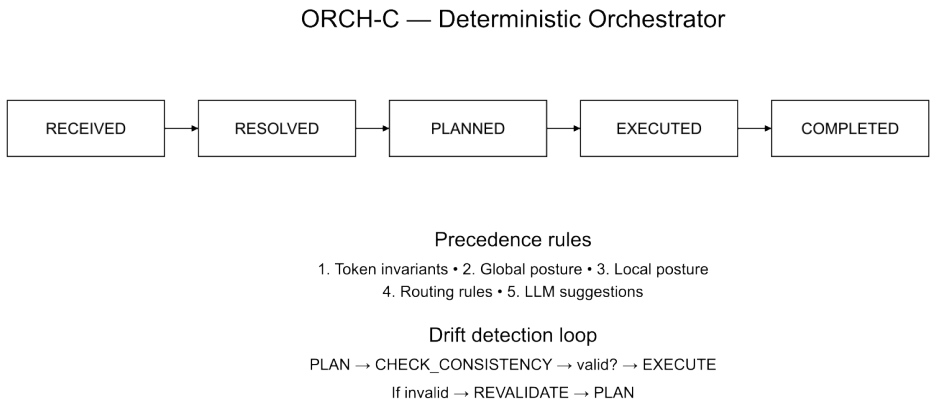

Figure 5: ORCH-C deterministic orchestration planner.

7 USS vs. Conventional LLM Systems

Figure 6 contrasts USS with conventional LLM pipelines.

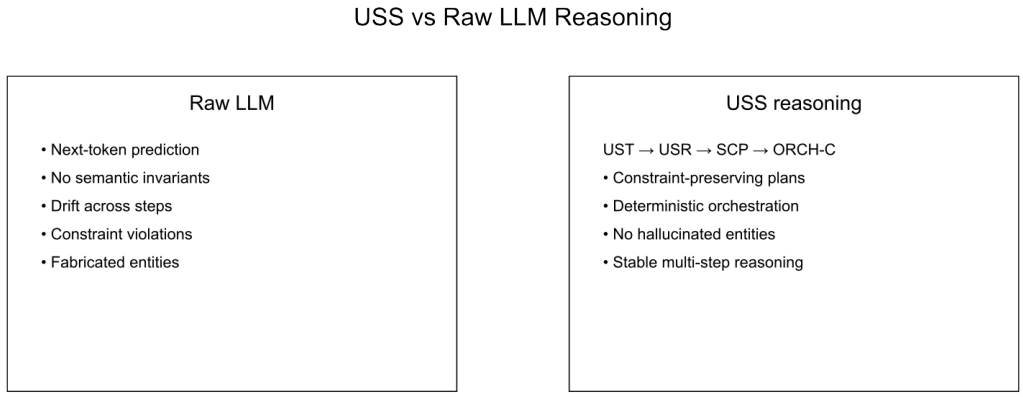

Figure 6: USS compared to conventional LLM systems.

Where LLMs operate probabilistically, USS enforces meaning as the primary computational object.

8 Related Work

A large body of work explores how to extend or constrain LLMs using external structure. Scaling laws [5] characterize how performance improves with model and dataset size but do not address semantic representation.

Tool-use approaches such as Toolformer [6] teach models to call external tools, while frameworks like ReAct [7] combine reasoning and acting in an interleaved loop. Constitutional AI [1] encodes alignment constraints as a higher-level rule set; SWE-Agent [3] demonstrates agentic code editing grounded in a software environment. LangChain [2] systematizes many of these patterns into a developer framework.

In parallel, classical automated planning [4] provides mature tools for explicit state-space search and plan synthesis, but these systems typically lack the open-ended generative capability of modern LLMs.

USS differs from these lines of work in two ways:

- It introduces a universal semantic token substrate (UST) that all modules share, rather than treating prompts or tool calls as opaque strings.
- It defines a deterministic runtime (USR), control protocol (SCP), and planner (ORCH-C) that together govern LLM behavior as one component inside a larger semantic system, rather than as the primary locus of intelligence.

USS can integrate with existing agent frameworks and planning systems, but its primary contribution is to treat meaning, not text, as the central computational object.

9 Example Execution Flow

To illustrate USS behavior, consider a user request:

> "Draft a cross-department strategy memo describing our transition to a semantic runtime."

Under USS, execution proceeds:

1. Caller/HIL Layer: Extracts domain, document type, audience, constraints.
2. UST Layer: Converts intent into typed semantic tokens.
3. SCP: Routes tokens through posture and directive logic.
4. ORCH-C: Produces a deterministic, multi-step execution plan.
5. Governance Layers: CAP and EIL enforce autonomy, ethics, and alignment.
6. Output Layer: Structured, validated semantic document.

This demonstrates how USS transforms a single query into a governed, auditable reasoning pipeline.

10 Conclusion

Universal Semantic Systems provides a meaning-first computational architecture for building deterministic, aligned, and semantically coherent AI systems. By introducing a semantic token substrate, a runtime with posture and invariant control, a structured protocol for directive routing, and a deterministic planner, USS transforms foundation models into semantically governed reasoning engines.

Future work includes empirical benchmarking, evaluation of semantic stability, and advanced meaning-formation modeling. USS is intended as a foundation for research on long-horizon reasoning and as an infrastructure layer for future alignment and governance systems.

References

[1] Yuntao Bai et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[2] Harrison Chase. The LangChain framework for building contextual AI agents. *Technical Report*, 2023.

[3] Steven Feng et al. SWE-Agent: Agent-computer interfaces for software engineering. *arXiv preprint arXiv:2310.06770*, 2023.

[4] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[5] Jared Kaplan et al. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[6] Timo Schick et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[7] Shunyu Yao et al. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.