Semantic Control Protocol (SCP): A Typed Orchestration Layer for Deterministic Language Systems

Robert Hansen
Design Logic

November 2025

Abstract

Modern large language models (LLMs) are powerful generators of text, but they lack a stable, machine-auditable notion of intent, state, and responsibility. Raw prompts act as untyped message strings, leaving orchestration, safety, and multi-step planning to ad hoc conventions. The *Semantic Control Protocol* (SCP) introduces a typed, schema-bound control layer that sits above natural language and below execution engines. SCP interprets human and agent intent into structured control directives, routes semantic operations, applies constraints, and orchestrates multi-step flows without modifying model internals. Combined with *Semantic Tokens* (STs)—a meaning-level substrate for content and state—SCP enables deterministic, auditable orchestration over probabilistic language models. This paper defines the core SCP model, its type system, routing semantics, and integration with STs, and outlines how SCP supports reliable multi-agent coordination in systems like LOS / SynCE.

1 Introduction

Large language models have made it practical to program with natural language, but they do not provide the same guarantees as typed programming languages or formal protocols. Prompts are opaque strings; control flow is implicit; safety checks and routing rules are scattered across bespoke code paths. As context windows grow and multi-agent architectures proliferate, the absence of a shared, governed control layer becomes a major source of drift, misalignment, and failure.

The Semantic Control Protocol (SCP) is a response to that gap. SCP is a semantic orchestration layer that:

- interprets human and agent intent into structured control directives;
- routes requests through a governed pipeline of tools, agents, and models;
- applies constraints and posture settings before generation, not after;
- maintains an auditable trace of what was requested, why, and how it was executed.

SCP is not another DSL for prompts; it is a control protocol that treats prompts and model outputs as one kind of payload among others, embedded in a typed envelope. When coupled with Semantic Tokens (STs) as a meaning substrate, SCP provides the control counterpart to STs' representation of content and state.

2 Background and Lineage

SCP is intentionally conservative: it reuses ideas from classical communication theory, formal semantics, and AI planning, but applies them to LLM orchestration.

Communication and channels. Shannon's theory of communication models messages transmitted over noisy channels with explicit encoding and decoding layers [7]. SCP borrows this separation: natural language is treated as an encoding of intent, while SCP messages form the structured channel through which that intent is carried and acted upon.

Meaning and formal semantics. Work in formal semantics and semiotics—including Peirce's triadic sign model [3], Tarski's definition of truth in formalized languages [8], and Kaplan's treatment of context and demonstratives [4]—demonstrates that meaning depends on both expression and interpretation rule. SCP assumes that STs provide a typed, schema-bound representation of meaning, while SCP defines how those representations are used to drive behavior.

Structure and generative systems. Chomsky's generative tradition [1, 2] frames language as a structured space of well-formed expressions. SCP does not attempt to replace the internal generative machinery of LLMs, but it mirrors this idea on the control side: only well-typed SCP messages are considered valid control instructions.

Agents and AI systems. Classical AI texts such as Russell and Norvig [5] describe agents as entities that perceive, reason, and act in an environment according to a performance criterion. SCP treats each agent—human or artificial—as a participant in a governed protocol, with explicit roles, permissions, and postures. It is designed to support multi-agent architectures where coordination cannot be left to informal conventions.

Intent, understanding, and limits. The Chinese Room argument [6] reminds us that symbol manipulation alone does not guarantee understanding. SCP does not claim to solve philosophical questions of understanding; instead, it offers an auditable link between the representations (STs) and the actions triggered from them, so that human operators can inspect and correct system behavior.

3 Problem Statement

Raw prompt-based orchestration exhibits several recurring problems:

1. **Unstructured control.** Control decisions (which tool to call, which agent to invoke, what constraints to apply) are embedded implicitly in long prompts or external glue code.
2. **Lack of type safety.** There is no enforced contract on what a model is allowed to return, or on what a controller is allowed to ask for.
3. **Weak auditability.** When a system misbehaves, it is difficult to reconstruct which instructions, modes, and constraints were in effect at each step.
4. **Multi-agent drift.** In agent swarms, conventions about message shape and semantics quickly diverge, leading to subtle incompatibilities and semantic drift.

5. **Runtime safety as an afterthought.** Safety filters often wrap the outside of the system, instead of being integrated into the core control loop.

SCP addresses these issues by making control explicit, typed, and logged.

4 Semantic Control Protocol Overview

At a high level, an SCP message is a typed envelope:

```
SCP_Message = { header, posture, routing, payload }
```

The *payload* may contain natural-language instructions, Semantic Tokens, tool calls, or other data. The header, posture, and routing fields allow ORCH-style orchestrators to reason about the message without inspecting its full textual content.

4.1 Core Concepts

Domains and scopes. Each SCP message belongs to a domain (e.g. `LOS.Core`, `SynCE.Public`, `Eden.R&D`) and scope (e.g. `session`, `workflow`, `tenant`). Domains control which routing rules and Binder policies apply.

Posture. Posture encodes the behavioral stance of the system—for example `FAST_HELPFUL`, `CAREFUL_EXPLAIN`, or `CAUTIOUS_LIMITED`. Posture influences which tools are eligible, how aggressively to optimize, and which safety margins to enforce. Posture is treated as a first-class field, not a vague style hint.

Constraints and budgets. SCP messages may carry explicit constraints: maximum steps, token budgets, latency targets, or safety levels. This allows deterministic enforcement of resource and risk boundaries, similar in spirit to planning horizons in classical AI [5].

Semantic Tokens. Where STs are present, SCP treats them as the canonical representation of meaning. Text is seen as a lossy surface; STs carry the durable semantics used for routing and Binder validation.

5 Type System

SCP uses a lightweight type system for messages and payloads:

- **Message types** (e.g. `QUERY`, `PLAN`, `EXECUTE`, `EVAL`) describe the role of a message in a workflow.
- **Payload types** describe what is carried: ST graphs, tool-call specifications, natural-language segments, or structured reports.
- **Capability types** describe what an agent or tool can handle, enabling routing based on type compatibility.

Types are intentionally simple: they are designed for easy implementation in JSON, Protobuf, or similar serializations. SCP does not require a specific representation; it requires only that the types be machine-checkable and versioned.

6 Execution Model

SCP assumes an orchestrator—such as ORCH-C in LOS / SynCE—that consumes SCP messages, plans an execution trace, and emits new SCP messages as results. A typical loop is:

1. **Ingress.** A human or agent submits an SCP message. The orchestrator validates schema, domain, and posture.
2. **Planning.** The orchestrator selects a plan template based on message type, domain, and posture. Budget and safety constraints are attached.
3. **Tool and agent routing.** Each step of the plan is turned into SCP sub-messages directed to tools or agents whose capability types match.
4. **Binder validation.** Binder-style validators check STs and SCP headers for alignment with global rules (e.g. GR–007 Truth Over Comfort).
5. **Aggregation and response.** Results from tools and agents are combined into a final SCP message containing both semantic artifacts (STs) and human-readable text.

Because every step flows through SCP envelopes, the entire execution trace is auditable: each decision can be inspected after the fact, including which posture, constraints, and validation rules were in effect.

7 Integration with Semantic Tokens

SCP and STs are complementary:

- STs represent *what* the system knows—concepts, entities, relationships, plans, and states—in a governed meaning layer.
- SCP represents *how* the system behaves—control decisions, routing, constraints, and postures.

In a SynCE-style stack, the integration looks like:

1. Natural-language input is parsed into candidate STs and an SCP `QUERY` message.
2. The orchestrator plans using SCP; STs provide durable references for entities and goals.
3. Tools and agents operate over STs where possible, using text only when interaction with humans or external black-box models is needed.
4. Final outputs include both ST updates and human-readable explanations.

This mirrors classical distinctions between representation and control in AI planning [5], but instantiated for LLM-based systems.

8 Evaluation Sketch

A full empirical evaluation is future work, but SCP suggests several measurable axes:

- **Drift under long-horizon workflows:** compare ST+SCP systems against prompt-only baselines on correctness and stability.
- **Audit time:** measure how quickly an operator can reconstruct "what happened" in a complex workflow.
- **Inter-agent compatibility:** measure how often independently-built agents interoperate successfully when constrained to SCP message types.

The goal is not raw benchmark scores, but *operational reliability*: the ability of a system to behave predictably and to surface its own reasoning in a way that human operators can inspect and govern.

9 Discussion

SCP does not attempt to resolve philosophical debates about understanding or consciousness [6]. Instead, it treats LLM-based systems as powerful pattern manipulators whose behavior must be constrained by explicit, auditable control structures. By placing a typed protocol between human intent and machine action, SCP offers a practical path toward systems that are:

- safer, because safety rules are integrated into the core control loop;
- more transparent, because every decision flows through a structured envelope;
- more composable, because agents and tools can coordinate via shared types and postures rather than ad hoc prompts.

10 Conclusion

The Semantic Control Protocol is a simple idea: treat control as a first-class, typed, semantic object rather than an emergent property of prompts and glue code. When combined with Semantic Tokens as a meaning substrate, SCP provides a coherent architecture for deterministic orchestration over probabilistic language models. The approach draws on long-standing work in communication theory, semantics, and AI agents [1–5, 7, 8], but is engineered for contemporary LLM ecosystems.

Future work includes formalizing the SCP type system, defining reference implementations for common domains (e.g. research assistants, multi-agent coding systems), and measuring the impact of SCP+ST architectures on long-horizon reliability and human trust.

References

[1] Noam Chomsky. *Syntactic Structures.* Mouton, The Hague, 1957.

[2] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, MA, 1965.

[3] Charles Hartshorne and Paul Weiss, editors. *Collected Papers of Charles Sanders Peirce, Vol. I–VI.* Harvard University Press, Cambridge, MA, 1931.

[4] David Kaplan. Demonstratives. In Joseph Almog, John Perry, and Howard Wettstein, editors, *Themes from Kaplan*, pages 481–563. Oxford University Press, 1989.

[5] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2010.

[6] John Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457, 1980.

[7] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[8] Alfred Tarski. The concept of truth in formalized languages. *Studia Philosophica*, 1933. English translation in *Logic, Semantics, Metamathematics*, Oxford University Press, 1956.