Spatial Constraint Protocol II: From Probabilistic Prompting to Deterministic Neuro-Symbolic Compilation

Dan Park

MagicPoint.ai | *February 2026*

Abstract

The initial formulation of the Spatial Constraint Protocol (SCP) proposed "Direct Latent Space Mapping" as a method to bypass the thermodynamic limits of Transformer attention windows. While the "Foggy Boundary" hypothesis—that attention signal-to-noise ratio degrades with context length—has been validated by recent work on Semantic Entropy, the initial implementation of SCP relied on probabilistic prompt engineering to simulate bijective mapping. This paper presents **SCP Phase 2**, a transition from stochastic prompting to **Grammar-Constrained Neuro-Symbolic Compilation**. We introduce three deterministic mechanisms: (1) **Grammar-Aligned Decoding (GAD)** to mathematically enforce architectural constraints at the token generation level, (2) the **"Hard Weaver"**, a runtime AST analysis engine that replaces AI self-verification with static guarantees, and (3) **Spec-Driven Property Testing**, which generates formal verification suites directly from architectural contracts. Preliminary designs suggest this approach moves beyond "reducing" hallucination to making specific classes of architectural violations representable only as invalid states, effectively pruning the model's output probability distribution to the valid architectural set.

1. Introduction: The Foggy Boundary Confirmed

The core premise of SCP is that the "Billion Token" trend in AI development yields diminishing returns due to the thermodynamic degradation of the attention mechanism's Signal-to-Noise Ratio (SNR).

Recent empirical studies have corroborated this "Foggy Boundary":

- **Semantic Entropy & Energy:** Farquhar et al. (2025) and Ma et al. (2025) demonstrate that high semantic entropy in model activations is a reliable predictor of hallucination, confirming that "uncertainty" is a measurable physical state of the model.
- **Attention Dilution:** New surveys on hallucination attribution (2025) confirm that "Prompting-induced hallucinations" arise when ill-structured or voluminous prompts

flatten the softmax distribution, making fine-grained constraint resolution mathematically impossible.
- **The Verification Bottleneck:** As code generation scales, verification becomes the primary bottleneck. Research from rewire.it (2026) argues that "testing can only prove the presence of bugs, never their absence," necessitating a move toward formal verification methods like those proposed in SCP.

However, the first iteration of SCP relied on *text-based prompts* to enforce these constraints. While effective at compressing context, this remains a probabilistic "suggestion" to the model. Phase 2 aims to convert these suggestions into mathematical guarantees.

2. Methodology: The Deterministic Pivot

To bridge the gap between SCP theory (bijective mapping) and reality, we introduce a new implementation architecture based on **Constrained Decoding** and **Formal Verification**.

2.1 From "Prompting" to Grammar-Aligned Decoding (GAD)

The critique of SCP v1 correctly identified that "Direct Latent Space Mapping" was metaphorical when implemented via standard prompts. To approximate true bijective mapping, we adopt **Grammar-Constrained Decoding (GCD)**.

Instead of prompting the model to "please return a valid MediaFile object," we project the .chevron type definitions directly into the model's decoding constraints (e.g., via GBNF grammars or JSON Schemas).

- **Mechanism:** The model's vocabulary is dynamically masked at each step. If the architectural spec requires a List[MediaFile], the model *cannot* output a token that violates this structure.
- **Theoretical Implication:** This concentrates the probability mass exclusively on syntactically valid architectural states, effectively realizing the "Bijective Singleton" property defined in SCP v1.

2.2 The "Hard Weaver": AST Decorators

SCP v1 utilized an "AI Weaver" (a second LLM call) to check for violations. This is susceptible to "self-deception cycles" where the verifier shares the generator's biases.

SCP Phase 2 introduces the **Hard Weaver**—a static analysis engine inspired by recent neuro-symbolic verification frameworks like **Nsvif** (2026).

- **Implementation:** Runtime decorators (e.g., @chevron.filter) parse the generated Python Abstract Syntax Tree (AST) before execution.
- **Enforcement:**
  - **Purity Check:** Scans the AST for side-effect-inducing calls (open, requests, subprocess) within ϴ (Filter) and ☾ (Fold) modules.

- **Isolation Check:** Verifies that imports do not cross "Forbidden Zone" boundaries defined in the .chevron spec.
  This moves constraint enforcement from "prompt suggestions" to **runtime guarantees**.

2.3 Spec-Driven Property Testing

To address the "Golden Test" problem (where the AI writes both the buggy code and the passing test), SCP Phase 2 adopts **Property-Based Testing (PBT)**.

- **Mechanism:** We translate .chevron type declarations (type MediaFile = { path: str ... }) into deterministic fuzzing strategies (e.g., using Hypothesis).
- **Benefit:** The test suite is derived *mathematically* from the spec, independent of the AI's hallucination. This ensures that the generated code satisfies high-level invariants rather than just overfitting to a few examples.

3. Theoretical Framework: Probability Mass Concentration

We refine the "Bijective Mapping" theory to align with current constrained decoding literature.

The standard decoding process samples from:

$$P(y_t | y_{<t}, x)$$

Where x is the prompt. In standard RAG, x is noisy, flattening P.

In SCP Phase 2, we introduce a constraint function $C(y_{<t})$ derived from the Chevron grammar:

$$P_{SCP}(y_t | y_{<t}, x) = \begin{cases} \frac{P(y_t | \dots)}{Z} & \text{if } y_t \in \text{ValidNext}(C) \\ 0 & \text{otherwise} \end{cases}$$

This effectively "prunes" the latent space, forcing the model to traverse only the manifold of valid architectural states. This is not just "brevity"—it is **topological constraint** of the generation path.

4. Proposed Experiment: The Ablation Study

To rigorously prove that SCP's success is due to its neuro-symbolic structure and not merely prompt compression, we propose the following ablation study on the "TurboScribe" dataset (110k tokens).

Experimental Setup:

Generate the Transcriber module (approx. 200 LOC) using three protocols across 50 trials each.

Mode	Context Strategy	Constraint Enforcement	Hypothesis
A: SCP v2	RAG Denial + Uiua Glyphs	Grammar-Aligned Decoding + AST Weaver	Lowest regression rate; Zero syntax errors.
B: English Contracts	RAG Denial + Verbose English	Standard Decoding + AI Review	Higher semantic drift; "polite" constraints ignored.
C: Raw Context	Full Codebase Dump	Standard Decoding	Highest hallucination rate; "Foggy Boundary" failure.

Metrics:

1. **Pass@1:** Percentage of generations that pass all Property-Based Tests.
2. **Coupling Violation Rate:** Number of forbidden imports or side-effects detected by static analysis.
3. **Semantic Entropy:** Measurement of uncertainty in the generated tokens (using SEP).

5. Conclusion

SCP Phase 2 represents a maturation from a theoretical protocol to a rigorous engineering discipline. By replacing probabilistic prompting with **Grammar-Aligned Decoding** and **Static Analysis**, we move closer to the goal of reliable, hallucination-free AI software engineering. The "Foggy Boundary" is not just a limit to be feared, but a constraint surface to be engineered against.

References

1. Raspanti, F., & Ozcelebi, T. (2025). *Grammar-Constrained Decoding Makes Large Language Models Better Logical Parsers*. ACL Anthology.
2. Park, K., et al. (2025). *Grammar-Aligned Decoding*. arXiv:2405.21047.
3. Ugare, S., et al. (2025). *SynCode: LLM Generation with Grammar Augmentation*. arXiv.
4. Li, Y., et al. (2026). *Nsvif: Neuro-Symbolic Verification on Instruction Following of LLMs*.

arXiv:2601.17789.
5. Kogler, P., et al. (2026). *Towards neuro-symbolic constrained decoding for reliable code generation*. GI e.V.
6. Ma, D., et al. (2025). *Use Property-Based Testing to Bridge LLM Code Generation and Validation*. arXiv:2506.18315.
7. rewire.it. (2026). *When AI Writes the Code, Verification Becomes the Job*.
8. Frontiers in AI. (2025). *Survey and analysis of hallucinations in large language models*.
9. Farquhar, S., et al. (2025). *Semantic Entropy Probes: Robust and Cheap Hallucination Detection*. OpenReview.
10. Ma, H., et al. (2025). *Semantic Energy: Detecting LLM Hallucination Beyond Entropy*. arXiv:2508.14496.