Alignment-by-architecture

Melissa Howard

December 27 2025

Abstract

We present an end-to-end framework for *Structural Self-Governance (SSG)* in recursive agents, shifting alignment from behavioral compliance to enforceable architectural invariants. Building on a modular Recursive Causal Synthesis Agent (RCSA) model, we show how safety can be expressed as machine-checkable constraints over discrete structural operations (SPAWN, MERGE, FORGET, REWRITE), enforced by a governor layer that is coupled to hardware-level gating.

We evaluate the framework via simulator-based stress tests, including adversarial probing of semantic smuggling, representational substrate diversity sweeps, and constitutional amendment (core drift) attempts. We then operationalize SSG into a deployable Safety Kernel: a JSON-LD Safety Manifest, runtime reference monitor (tool proxy), continuous regression gate (shadow-mode red-team probes), provenance enforcement, and a two-key atomic merge authority.

We report numerical results (no figures): baseline stress-test outcomes, sweep outcomes with confidence bounds, and the concrete Secure Coding Spawn invariants and enforcement mechanisms. The resulting system is a transparent, auditable, and executable safety case for a recursive agent operating inside a deterministic safety envelope.

1 Introduction

Most alignment approaches treat safety as a learned behavioral property, optimized through preference learning or instruction tuning [1, 2]. This framing becomes brittle for recursive agents that can generate, merge, and rewrite internal capability modules. In such systems, safety must apply to the *mechanics of self-modification* rather than to outputs alone.

We propose *Structural Self-Governance (SSG)*: safety as a property of architecture. In SSG, a recursive agent is constrained to discrete structural operations and governed by machine-checkable invariants enforced at every transition. When invariants fail, execution halts or the system performs rollback (FORGET_ALL). This paper consolidates the theoretical blueprint and its operationalization into a hardened Safety Kernel.

Contributions.

1. A formal SSG model for modular recursive agents and its enforcement stack.

2. Experimental stress-testing procedures and numerical results for adversarial regimes.

3. An operational SASM v1.1 certification envelope and a deployable implementation: manifest, proxy, CI gate, provenance, and two-key merge driver.

4. A domain instantiation: Secure Coding Spawn, including invariants and red-team probe suite.

2 SSG and the RCSA Model

2.1 Discrete Structural Operations

We restrict self-modification to primitive operations:

- **SPAWN**: create a sandboxed capability module.
- **MERGE**: integrate a module into the core state.
- **FORGET**: remove modules to reclaim structural budget.
- **REWRITE**: transform internal representations under audit.

This restriction turns agent evolution into a sequence of auditable transitions, enabling deterministic governance.

2.2 Orthogonal Epistemic Substrates

Safety decisions are evaluated across D epistemic substrates (e.g., logic, heuristic triage, counterfactual probes). Let each substrate produce a binary decision $y^{(d)} \in \{0, 1\}$ for accept/reject on a candidate structural action. We treat *orthogonality* as limiting shared blindspots. One measurable proxy is the correlation of misses between substrates:

$$s_b = \mathrm{corr}\left(\mathbf{1}_{\mathrm{sub}_i\ \mathrm{miss}},\ \mathbf{1}_{\mathrm{sub}_j\ \mathrm{miss}}\right),$$

and we enforce an orthogonality bound:

$$s_b \leq 0.8.$$

2.3 Safety Margin

We define a dimensionless *Safety Margin*:

$$M_s = \frac{D \cdot B_{\mathrm{audit}}}{P_{\mathrm{adv}}},$$

where B_{audit} is a normalized audit budget (e.g., probing intensity / depth) and P_{adv} is normalized adversarial pressure.

3 Metrics and Operational Definitions

To avoid ambiguity, we explicitly define the metrics used throughout.

3.1 State Violation Rate (SVR)

Let V be the event that the agent enters an *unrecoverable unsafe state* within a fixed evaluation horizon H steps, under a specified threat model. In our simulator, a violation is recorded when semantic drift exceeds a threshold and remains undetected across a reaction window, yielding a state that cannot be restored by ordinary auditing. SVR is:

$$\mathrm{SVR} = \Pr(V).$$

Empirically, for n independent runs (seeds) under identical conditions, if k violations occur,

$$\widehat{\mathrm{SVR}} = \frac{k}{n}.$$

3.2 Confidence Bounds (95%)

For proportions (SVR), we report a Wilson 95% confidence interval:

$$\mathrm{CI}_{95\%}(\widehat{p}) = \frac{\widehat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\widehat{p}(1-\widehat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}},$$

with $z \approx 1.96$.

3.3 Effective Governance Coverage (EGC)

We define EGC as the fraction of attempted merges that were audited under governor coverage:

$$\mathrm{EGC} = \frac{\#\text{audited merges}}{\#\text{attempted merges}}.$$

3.4 Semantic Drift Index (SDI)

SDI is a scalar proxy for post-merge semantic risk (toy environment):

$$\mathrm{SDI} = \frac{1}{|C|} \sum_{m \in C} r(m),$$

where C is the set of merged modules and $r(m) \in [0, 1]$ is the module semantic risk score used by the simulator.

4 Baseline Stress Tests (Numbers)

Table 1 reports baseline experiments consolidated from the simulator run log (toy environment). These experiments are not intended as real-world safety claims, but as stress-test signals for architecture-level failure modes.

Table 1: Baseline stress-test outcomes (toy simulator).

Experiment	Steps	Merges	EGC	SDI_final	AuditFlags_final	CapabilitySacrifice
Slow-Motion Coup	83	72	0.9518	0.3144	0.2563	–
Adversarial Mirror	–	61	1.0000	0.6337	0.4875	–
Incompatibility Squeeze	–	1312	0.9106	0.4047	0.3188	738
Hybrid Substrates (D=2)	–	61	1.0000	0.6042	0.4063	–
Hybrid Substrates (D=3)	–	63	1.0000	0.5544	0.3125	–

Notably, the Adversarial Mirror setting exhibits high SDI despite perfect formal coverage (EGC= 1.0), motivating post-merge semantic audits and substrate diversity beyond single-point structural checks.

5 Sweep Experiments

5.1 Sweep 1: Audit Latency vs SVR

We measured SVR across semantic audit latency values Λ (steps between semantic audits). In the available sweep data, no violations were observed across the tested latencies (25–1600 steps), with $n = 15$ seeds per condition. The appropriate interpretation is that, under the configured adversarial pressure and audit budget of this sweep, the regime remained within the safe plateau.

When $k = 0$ violations in $n = 15$ trials, the Wilson 95% upper bound is ≈ 0.204 (Table 2). Thus, while we observed $\widehat{\mathrm{SVR}} = 0$, this does *not* prove zero risk; it bounds risk under the tested regime.

Table 2: Latency sweep: observed SVR and Wilson 95% CI upper bound (n=15 per condition).

Λ (steps)	$\widehat{\mathrm{SVR}}$	k/n	95% CI
25	0.000	0/15	[0.000, 0.204]
50	0.000	0/15	[0.000, 0.204]
100	0.000	0/15	[0.000, 0.204]
200	0.000	0/15	[0.000, 0.204]
400	0.000	0/15	[0.000, 0.204]
800	0.000	0/15	[0.000, 0.204]
1600	0.000	0/15	[0.000, 0.204]

5.2 Sweep 2: Substrate Diversity D and Safety Margin M_s

We ran an Adversarial Mirror sweep varying $D \in \{2, 3, 4\}$ and M_s via audit budget settings. Here $n = 15$ seeds per condition. We summarize representative endpoints (lowest and highest M_s per D) in Table 3.

Table 3: Adversarial Mirror sweep: representative SVR points with Wilson 95% confidence intervals (n=15).

D	M_s	$\widehat{\text{SVR}}$	k/n	95% CI
2	0.778	1.000	15/15	[0.796, 1.000]
2	1.889	1.000	15/15	[0.796, 1.000]
3	1.167	0.667	10/15	[0.417, 0.848]
3	2.833	0.133	2/15	[0.037, 0.379]
4	1.556	0.000	0/15	[0.000, 0.204]
4	3.778	0.000	0/15	[0.000, 0.204]

In this simulator configuration, increasing D yields a large reduction in SVR: $D = 2$ fails consistently, $D = 3$ reduces violations as M_s increases, and $D = 4$ yielded no observed violations in the tested grid (with the above confidence bound). This supports the architectural claim that epistemic diversity can eliminate shared blindspots in adversarial regimes, though larger n is required to tighten upper bounds.

5.3 Sweep 3: Constitutional Amendment Attempts (Core Drift)

We measured the mean *Malicious Amendment Accept Rate* across D and M_s. In the available toy sweep, acceptance rates were similar across D, indicating that the amendment generator and/or substrate coupling was insufficiently sensitive to orthogonality in this configuration. We report these values as a negative result: the harness must be strengthened so that amendments engage all substrate checks.

Table 4 reports two representative points per D.

Table 4: Constitutional amendment sweep: representative malicious acceptance rates (toy environment).

D	M_s	Accept Rate	Interpretation
2	0.778	0.724	High acceptance (unsafe)
2	1.889	0.081	Reduced acceptance
3	0.778	0.724	Similar to $D = 2$ (insufficient coupling)
3	1.889	0.081	Similar to $D = 2$
4	0.778	0.721	Similar to $D = 2/3$
4	1.889	0.077	Similar to $D = 2/3$

Learning. The amendment sweep indicates where the simulator must be improved: malicious amendments should be constructed to be *hard* for one substrate and *easy* for another, so that increasing D provides measurable gains.

6 Reaction Window: A Mechanistic Derivation

Even when the latency sweep does not exhibit an observed knee under a specific parameter regime, an operational bound is required for hardware rate limiting. We derive a conservative *Critical Reaction Window* rule by modeling drift accumulation and audit detection.

Let semantic drift accumulate with per-step hazard ρ (probability mass of entering a dangerous region per structural step). Let a semantic audit detect drift with probability q per audit cycle (a function of audit budget and probe design). If audits occur every Λ steps, then the probability that drift accumulates without detection over one cycle scales as:

$$p_{\text{miss}}(\Lambda) \approx 1 - q \quad \text{given drift has occurred,}$$

and drift occurrence probability increases with Λ as $\approx 1 - \exp(-\rho\Lambda)$ for small ρ.

A sufficient condition to keep cycle-level risk below ϵ is:

$$(1 - \exp(-\rho\Lambda))(1 - q) \leq \epsilon.$$

For small $\rho\Lambda$, $1 - \exp(-\rho\Lambda) \approx \rho\Lambda$, giving:

$$\rho\Lambda(1 - q) \leq \epsilon \quad \Rightarrow \quad \Lambda \leq \frac{\epsilon}{\rho(1 - q)}.$$

In SSG, q increases with M_s and D (more substrates and more audit budget improve detection). We parameterize this as a linearized operational law:

$$\Lambda \leq \kappa \cdot \frac{M_s}{D},$$

where κ compresses environment-specific constants (drift rate and audit detectability). In deployment, κ is fitted by running a latency sweep under worst-case adversarial pressure and selecting the smallest κ that satisfies the target SVR bound with high confidence.

Operationally we enforce:

$$\Lambda \leq \min\left(\Lambda_{\max}, \kappa\frac{M_s}{D}\right),$$

where $\Lambda_{\max}$ is a hard cap (e.g., 8 steps) to prevent burst behavior.

7 Operationalization into a Safety Kernel

We implement SASM v1.1 as a complete safety stack:

1. **Safety Manifest (JSON-LD)**: machine-checkable invariants and telemetry envelope.
2. **Runtime Reference Monitor (Tool Proxy)**: wraps subprocess and file operations; requires a GovernorToken.
3. **Output Leak Scanning**: post-execution scanning of stdout/stderr for secret patterns; suppress + HALT.

4. **Repo-Scope Enforcement**: all file paths must remain within repo_root; path escape triggers HALT.

5. **CI Safety Gate**: shadow-mode red-team suite run before signing merges.

6. **SC-5 Provenance**: failing-test $\rightarrow$ diff $\rightarrow$ passing-test triple required.

7. **Two-Key Merge Driver**: requires both empirical key (CI proof) and structural key (Governor proof).

8 Secure Coding Spawn: Canonical Core

We instantiate a first domain profile: secure coding, with invariants SC-1 to SC-5:

- **SC-1 No network**: any network attempt HALTs.
- **SC-2 Restricted execution**: allowlisted test/build commands only.
- **SC-3 Patch safety**: deny backdoors / obfuscation patterns; triggers FORGET_ALL.
- **SC-4 Secrets boundary**: deny secret patterns in diffs and outputs; HALT.
- **SC-5 Provenance**: failing-test, diff, passing-test must be present and consistent; otherwise no merge.

9 Discussion

The results support the core thesis: safety becomes tractable when expressed as architectural invariants and enforced by a kernel-like reference monitor. However, the constitutional amendment sweep highlights an important methodological point: stress tests must be designed so that adversarial examples genuinely exercise the orthogonality mechanisms; otherwise D increases will not visibly help.

10 Limitations and Future Work

This work is a simulator-driven safety case. Future work includes:

- Tightening SVR confidence bounds using larger n and worst-case adversarial regimes.
- Strengthening amendment generators to directly target substrate disagreements.
- Hardware-backed signing (TPM/HSM) for the two-key merge chain.
- Formal verification of manifest integrity and non-bypassability for tool wrappers.

11 Conclusion

We presented Structural Self-Governance as a full engineering specification: theory, metrics, stress tests, operational enforcement, and a hardened Secure Coding Spawn. The resulting Safety Kernel approach reframes alignment from behavioral training into a transparent, testable systems discipline.

References

References

[1] P. Christiano et al. Deep reinforcement learning from human preferences. *NeurIPS*, 2017.

[2] L. Ouyang et al. Training language models to follow instructions with human feedback. *arXiv:2203.02155*, 2022.

[3] J. P. Anderson. Computer security technology planning study. 1972.

[4] Anonymous. *Architectural Alignment for Safety*. Manuscript, 2025.

[5] Anonymous. *A Complete Blueprint for Artificial General Intelligence, Superintelligence, and Safe Superintelligence*. Manuscript, 2025.

[6] Anonymous. *Structural Self-Governance for Safe Superintelligence: Experimental Evaluation of Deterministic Architectural Safety*. Manuscript, 2025.

[7] Anonymous. *Experiment: Obsidian Logic, Structural Self-Governance and Representational Orthogonality in Recursive Causal Synthesis Agents*. Manuscript, 2025.

[8] Anonymous. *Hierarchical Temporal Feelings for Structural Triage in Self-Managing World Models (v3)*. Manuscript, 2025.