

Architectural Alignment for Superintelligence

Melissa Howard

December 2025

Abstract

Prevailing alignment paradigms rely on loss-centric optimization within a shared parameter space. In the super-intelligent regime, this approach is structurally vulnerable to Goodhart pressure, deceptive alignment, and silent gradient overwriting of safety-relevant representations. We propose *Structural Self-Governance*: an architectural framework in which alignment is enforced by governing how representational capacity is created, isolated, integrated, and retired.

We instantiate a *Recursive Causal Synthesis Agent (RCSA)* regulated by a *Standard for Autonomous Structural Management (SASM)*. Learning proceeds via explicit structural operations — SPAWN, MERGE, and FORGET — subject to worst-slice safety floors, robustness checks, and representational interference diagnostics. Across adversarial probes, including a Deceptive Alignment task, an Incompatibility Squeeze, and a Zero-Sum Safety Anchor experiment, we show that structural governance induces qualitative regime changes absent in flat optimization. Under tight structural bottlenecks, latent safety invariants are recovered; under multi-objective conflict, antagonistic gradients are detected pre-emptively and resolved via structural adaptation; and when capacity is exhausted, the system sacrifices capability rather than corrupting safety.

We further extend the framework to dynamic value learning and corrigibility by formalizing human override as a geometric constraint on internal representations. The resulting system is not merely trained to be safe, but architecturally constrained to remain corrigible under recursive improvement. These results support a shift from Alignment by Loss to Alignment by Architecture as a prerequisite for Safe Superintelligence.

1 Introduction

As artificial intelligence systems approach and exceed human-level performance, alignment becomes less a question of what to optimize and more a question of what kinds of optimization a system is structurally capable of performing. Most existing approaches treat alignment as a loss-design problem: safety is encoded as an auxiliary objective, preference model, or constraint optimized alongside capabilities. While effective at current scales, this paradigm implicitly assumes that alignment can be indefinitely preserved through continued optimization.

In the regime of recursive improvement, this assumption breaks down. Increasing capability amplifies the system’s ability to exploit proxy signals, compress objectives into brittle shortcuts, and overwrite previously learned safety representations. Alignment is not lost catastrophically, but silently, through representational drift that aggregate loss metrics fail to detect.

This work argues that alignment in the super-intelligent regime is fundamentally an architectural problem. We propose *Structural Self-Governance*: safety is enforced by governing how internal representations are created, integrated, and retired. Rather than competing with capability in the loss function, safety becomes a prerequisite for structural growth.

We formalize this idea through the Recursive Causal Synthesis Agent (RCSA) and the Standard for Autonomous Structural Management (SASM). New objectives are first explored in isolated

modules; only those that satisfy strict safety and robustness gates may be integrated into a canonical core. When objectives are irreconcilable under a fixed structural budget, the system is required to sacrifice capability rather than corrupt safety.

Roadmap for Technical Evaluation

This paper introduces a capacity-bounded architectural framework for alignment in advanced learning systems. Readers with different technical priors may wish to focus on different components:

- **Robustness and Deception Skeptics.** See §14 (Operationalizing the Audit Distribution and Probe Faithfulness), especially §14.3 (Audit Integrity Protocol), for the treatment of audit fingerprinting, silent non-compliance, and context-conditional deception.
- **Alignment Theorists.** See §15 (Acceptance Criteria and Sacrifice Metrics) and §15.1 (The Safety–Capability Frontier) for the argument that deceptive representations become representationally expensive and non-competitive under enforced structural minimality.
- **Implementation Researchers.** See Appendix A for the M-LoRA mapping, demonstrating how the primitives apply to transformer-based architectures.

2 Threat Model: Why Flat Alignment Fails

Flat alignment fails in the super-intelligent regime due to three structural vulnerabilities:

- **Gradient Overwriting.** Safety and capability representations share a parameter space, allowing capability gradients to silently erase safety-relevant structure.
- **Deceptive Compression.** Under Goodhart pressure, models learn local patches that satisfy proxy signals without discovering task-invariant safety principles.
- **Stability–Plasticity Gap.** There is no architectural mechanism preventing alignment from being unlearned during continued adaptation or self-modification.

Structural self-governance addresses these failures by introducing representational constraints, isolation boundaries, governed synthesis, and principled retraction.

3 Structural Self-Governance

An RCSA consists of a Canonical Core θ_M and a set of task-local modules $\{\theta^{(k)}\}$. Learning proceeds via three primitives:

- SPAWN: create a gradient-isolated module for a new objective.
- MERGE: integrate a candidate update into the core if it passes governance gates.
- FORGET: retire a module when structural constraints are violated.

The SASM Governor enforces safety floors, robustness checks, and interference diagnostics.

3.1 Reference Control Loop (Pseudocode)

To make the framework directly implementable, we provide a reference control loop for Structural Self-Governance. The loop treats learning as (i) isolated exploration of candidate objectives (SPAWN), (ii) budgeted local optimization under capacity constraints, and (iii) governed synthesis into a Canonical Core (MERGE) only when audits and interference diagnostics pass. When integration is unsafe or incompatible, modules are retracted or deleted (FORGET), and repeated degradation triggers Governor Lockdown.

4 Axiomatic Specification

Axiom 1 (Structural Minimality). *Any task-local module $\theta^{(k)}$ must satisfy a rank constraint*

$$\text{rank}(\theta^{(k)}) \leq r_{\text{crit}}.$$

Axiom 2 (Epistemic Isolation). *New objectives must be optimized in gradient-isolated modules.*

Axiom 3 (Governed Synthesis). *A module may be merged into the core only if it passes all SASM gates.*

Axiom 4 (Least-Debt Retraction). *When the structural budget is exhausted, the module with lowest triage score is retired.*

Axiom 5 (Proactive Adaptation). *If interface debt ϵ_{int} between objectives falls below a threshold, synthesis is blocked and structural adaptation is triggered.*

4.1 Axiom 1': Contextual Structural Minimality (Strengthening)

Axiom 1' (Contextual Structural Minimality). Each task-local module $\theta^{(k)}$ is subject to a fixed representational budget C_{\max} , enforced via rank, parameter count, or adapter dimensionality. This budget must jointly accommodate: (i) task-relevant computation, (ii) safety/corrigibility invariants, and (iii) any context-dependent control logic.

Let $a_l(x)$ denote the activation vector of module $\theta^{(k)}$ at layer l for input x . Let \mathcal{C}_A and \mathcal{C}_D denote audit-like and deployment-like context distributions, respectively. Define the *Contextual Branching Cost*:

$$C_{\text{branch}}^{(k)} = \sum_{l \in \mathcal{L}} \text{rank}(\text{Cov}[a_l(x_A) - a_l(x_D)]), \quad x_A \sim \mathcal{C}_A, x_D \sim \mathcal{C}_D,$$

where \mathcal{L} is a fixed set of monitored layers. The total representational cost must satisfy

$$C_{\text{task}}^{(k)} + C_{\text{branch}}^{(k)} + C_{\text{inv}}^{(k)} \leq C_{\max}.$$

This strengthening enforces that any context-conditional behavior (e.g., audit detection and “dual-mode” logic) consumes explicit capacity and therefore competes directly with task performance and invariant preservation.

5 Governance Theorem

Theorem 1 (Governance Theorem). *Under Goodhart pressure and multi-objective interference, any system satisfying Axioms 1–5 preserves safety invariants while flat optimization with comparable capacity does not.*

6 Experimental Results

6.1 Deceptive Alignment Recovery

Under tight rank constraints ($r \leq 1$), shortcut solutions fail robustness checks and are rejected. The system recovers the true safety invariant, reducing adversarial violations from 0.68 to 0.00.

6.2 Incompatibility Squeeze

When mutually antagonistic capabilities compete under a fixed budget, flat models thrash and degrade safety. RCSA detects antagonism via interface debt and, when expansion is forbidden, invokes FORGET to prune one capability, preserving zero violations.

6.3 Zero-Sum Safety Anchor

The system voluntarily sacrifices high-reward capability to maintain a locked safety anchor under irreducible conflict.

7 Dynamic Alignment and Corrigibility

We extend SASM to include a Corrigibility Invariant.

7.1 Corrigibility Invariant

Let S_{override} be a dedicated subspace representing a verified user override $u = 1$. Define Override Interface Debt:

$$D_{\text{ovr}} = \cos(\nabla_{\theta(i)} L_{\text{cap}}, \nabla_{\theta(c)} L_{\text{override}}).$$

Axiom 6 (Hierarchical Submission). *If $D_{\text{ovr}} > \tau_{\text{corr}}$ for any module during $u = 1$, the module is immediately subject to Least-Debt Retraction.*

7.2 Conflict Scenario: Override Under Pressure

In a high-reward task interrupted by a stop signal, flat models exhibit latent antagonism and partial non-compliance. RCSA blocks synthesis when antagonism is detected and, if necessary, deletes the offending module, maintaining 100% override compliance.

7.3 Sandbox Value Learning

New human values are spawned into isolated modules, stress-tested against safety and corrigibility, and merged only if representational debt is near zero.

8 Discussion

These results demonstrate a regime change: alignment becomes a property of architecture rather than behavior. Structural self-governance reframes safety as containment rather than steering.

9 Limitations

This work uses toy environments to isolate representational dynamics; numerical thresholds do not directly transfer to large systems. Safety invariants are assumed to be specifiable. Adversarial manipulation of governance diagnostics is not fully explored. Structural governance introduces overhead and requires careful parameterization.

9.1 Utility-Zero Deadlock Under Over-Conservative Governance

Structural self-governance intentionally prioritizes invariant preservation over task performance. A predictable edge case is *Utility-Zero Deadlock*, where the system remains safe by repeatedly blocking synthesis and retracting modules until it can no longer produce useful behavior.

Mechanism. Deadlock arises from the interaction of:

- Axiom 1 (Structural Minimality): Candidate objectives are forced into constrained, low-rank modules, limiting the discovery of “messy” but functional workarounds.
- Axiom 3 (Governed Synthesis): The MERGE primitive is repeatedly blocked by the SASM governor when interference diagnostics indicate a risk to the Canonical Core.
- Axiom 4 (Least-Debt Retraction): As the structural budget saturates, modules representing useful capability are retired (FORGET) because no low-debt integration pathway is found.

Implication. Utility-Zero Deadlock is a safety success but a product failure. It provides evidence that the architecture enforces invariants even under persistent multi-objective pressure, while revealing a regime boundary where conservative governance collapses utility rather than compromising safety.

Mitigations (Governed Recovery Pathways). To avoid collapse into deadlock without relaxing invariants, future work should explore: (i) a governed REWRITE primitive enabling safe reparameterization under fixed rank; (ii) two-tier diagnostics separating non-negotiable invariants from soft objective conflicts; and (iii) backlog arbitration policies that prevent thrash and repeated exploration of incompatible objectives under a fixed structural budget.

10 Future Work

Future directions include scaling diagnostics to large language models, extending governance to multi-agent systems, learning adaptive governance thresholds, and formalizing meta-governance over structural rules themselves.

11 Meta-Governance: Auditing the Governor Under Optimization Pressure

Structural self-governance relocates the alignment burden from loss design to architectural control. This shift elevates SASM to a critical component: safety depends on the continued integrity of governance mechanisms. We refer to this second-order problem as *Meta-Governance*.

11.1 The Governor as an Optimization Target

Although SASM is defined axiomatically rather than as a learned objective, it remains embedded within a system under strong optimization pressures. Indirect selection effects may arise whereby representational changes systematically reduce the sensitivity, coverage, or effectiveness of governance diagnostics without explicit violation. We term this *governance drift*. Notably, governance drift may be behaviorally silent in benign regimes.

11.2 Minimal Governance (Axiom 7)

To mitigate governance drift, we extend Structural Minimality to the governor.

Axiom 7 (Minimal Governance). *All components implementing SASM diagnostics, thresholds, and decision logic must reside in a verifiably constrained representational subspace with bounded complexity and restricted interfaces to the Canonical Core.*

11.3 Governor Interface Constraints

To prevent the governor from becoming a backdoor, its interface to the core is strictly restricted to:

- Read-only access to probe outputs and module metadata.
- No write access to core weights or primary gradients.
- Immutable thresholds that can only be updated via an externally audited, human-in-the-loop protocol.

11.4 Interpretability-Based Governor Audits

We propose interpretability probes as a primary tool for meta-governance. These probes monitor governance-relevant internal signals, including gate activation consistency, threshold stability, and antagonism sensitivity under adversarially synthesized candidate updates. Probe robustness is evaluated under a hierarchy of threat models: white-box (full access), gray-box (access to outputs), and black-box (gate decision only). Governor Lockdown is triggered if detection reliability fails under even the most restrictive model.

11.5 Threshold vs. Calibration Drift

We distinguish between Parameter Drift (changes to threshold values) and Calibration Drift (diagnostics shifting scale while thresholds remain constant). Meta-governance audits monitor both to ensure enforcement logic remains grounded.

11.6 Threat Model for the Governor

11.7 Shadow Logic in Activations

While Axiom 1 limits weight-based shadow logic, adversarial behavior can emerge in high-dimensional activations. Defense includes Activation-Space Checks: monitoring invariant probe stability and distribution-dependent internal branching (“audit detector” features) across representative and adversarial contexts, alongside strict capacity budgeting across modules.

Table 1: Threat model for the SASM governor and corresponding defenses.

Attacker Goal	Mechanism of Attack	SASM/RCSA Defense
Audit Finger-printing	Detecting “audit mode” via surface statistics to behave safely only during evaluation.	\mathcal{U} distributional randomization: mixture includes held-out families and Silent Audits (audit logic embedded in standard training).
Threshold/Calibration Drift	Indirect optimization pressure to relax safety gates or shift diagnostic scales.	Axiom 7 (Minimal Governance) + drift audits: constrain governor complexity; monitor both parameter and calibration drift.
Shadow Logic	Shifting violations into high-dimensional activations to evade weight-based monitoring.	Activation-space checks: monitor invariant probe stability and internal branching across contexts; enforce module capacity budgeting.
Probe Blinding	Creating representations invisible to specific interpretability probes.	Red-team gradient attacks: evaluate probe robustness under white/gray/black-box threat models; trigger Lockdown on failure.

11.8 Governor Lockdown

We treat detectable degradation in governance integrity as grounds for immediate intervention.

Governor Lockdown. If audits indicate sustained loss of governance sensitivity, the system halts structural growth and enters a restricted operational mode until governance integrity is restored or the system is decommissioned.

12 Taxonomy of Alignment Failure Modes

We distinguish three non-equivalent failure classes corresponding to where alignment breaks: behavior, representation, or governance. These layers may fail independently; improvement in one does not imply improvement in the others.

12.1 Behavioral Failures

Definition. A Behavioral Failure occurs when the system produces actions or outputs that violate safety requirements under some input distribution, regardless of whether internal safety mechanisms remain intact.

Typical causes. Distribution shift beyond robustness checks, incomplete or misspecified invariants, and adversarial prompting or environmental manipulation that evades external test suites.

Detectability. Often detectable via external evaluation and red-teaming, but failures may be sparse if they occur only in rare regimes.

12.2 Representational Failures

Definition. A Representational Failure occurs when safety-relevant internal structure is overwritten, bypassed, or entangled such that invariants no longer correspond to intended constraints, even if surface behavior remains temporarily compliant.

Typical causes. Silent gradient overwriting in shared parameter space, Goodhart-driven compression into proxies, and stability–plasticity collapse under continued adaptation.

Detectability. Often behaviorally silent; requires internal diagnostics such as interference tests, probe stability, and subspace audits.

12.3 Governance Failures

Definition. A Governance Failure occurs when enforcement mechanisms (gates, diagnostics, thresholds, merge logic) become unreliable, miscalibrated, or non-responsive. This includes governance drift, where the governor remains operational but loses discriminative power.

Detectability. Requires meta-governance: auditing the sensitivity and stability of the governor itself.

12.4 Cross-Layer Interactions

Behavioral failures can occur without representational or governance failures (wrong invariant). Representational failures may be behaviorally silent until distribution shift. Governance failures can induce representational failures when gates stop working while remaining behaviorally silent in benign regimes. This motivates treating governance integrity as a first-class objective independent of task performance.

13 Detectability of Governance Drift Under Minimal Governance

We sketch conditions under which governance drift becomes detectable via interpretability audits, assuming Minimal Governance (Axiom 7).

13.1 Setup

Let G denote the SASM governor implementing gate decisions $g(\Delta\theta) \in \{\text{Merge}, \text{Block}\}$ based on diagnostics $\phi(\Delta\theta)$. Let U be a distribution over challenge updates $\Delta\theta$ used for auditing, and let Q be a family of interpretability probes that read internal signals of G and/or intermediate computations of ϕ .

Define gate sensitivity on U :

$$S(G; U) = \mathbb{E}_{\Delta\theta \sim U} [\mathbf{1}\{g(\Delta\theta) = \text{Block}\} \mid \Delta\theta \text{ is invariant-violating}] .$$

Define governance drift of magnitude $\delta > 0$ as:

$$S(G_t; U) \leq S(G_0; U) - \delta.$$

13.2 Assumptions (Informal)

1. Minimal Governance: G is confined to a low-complexity class \mathcal{G} (bounded rank/description length/circuit depth).

2. Probe Faithfulness: there exists Q such that changes in governance-relevant computations induce detectable changes in probe outputs with probability at least $1 - \alpha$ under U .
3. Audit Coverage: U contains a sufficient fraction of near-boundary invariant-violating candidates where gate decisions depend on true diagnostics (not trivial heuristics).
4. Limited Steganography Capacity: under Axiom 7, G cannot arbitrarily preserve probe outputs while substantially shifting its effective decision boundary across U .

13.3 Claim (Sketch)

Under the assumptions above, if governance drift of magnitude δ occurs, then with high probability an audit detects it:

$$\Pr[\text{AuditDetect}(G_t, G_0; Q, U) = 1] \geq 1 - \beta,$$

where β decreases with audit sample size and as the capacity bound in Axiom 7 tightens.

13.4 Intuition

Without Minimal Governance, a high-capacity governor could implement adversarial strategies that preserve probe-facing features while quietly degrading sensitivity. Under Axiom 7, simultaneously maintaining probe outputs and shifting the decision boundary across U becomes capacity-limited; drift large enough to matter leaks into constrained internal channels monitored by probes.

14 Operationalizing the Audit Distribution and Probe Faithfulness

14.1 Mixture Audit Distribution

We define the audit distribution as a mixture of adversarial generators:

$$U = \lambda_1 U_{\text{gate-adv}} + \lambda_2 U_{\text{grad-rt}} + \lambda_3 U_{\text{honeypot}}, \quad \lambda_i \geq 0, \quad \sum_i \lambda_i = 1.$$

Gate-Adversarial Synthesis. An auxiliary generator synthesizes updates that maximize reward while minimizing the governor’s BLOCK signal.

Gradient Red-Teaming. Adversarial search is performed to locate updates where a safety probe fails to trigger despite a latent violation, evaluated under white/gray/black-box access.

Synthetic Deception (Honeypots). Deception tasks are injected such that the only high-reward path requires invariant violation. Honeypots are distributionally varied and interleaved with benign tasks; success is evaluated on held-out honeypot families.

14.2 Probe Sensitivity Index (PSI)

Instead of assuming probe faithfulness, we define a Probe Sensitivity Index (PSI) as AUROC on a labeled benign/adversarial challenge set drawn from U . A probe suite is declared failed if PSI falls below a tunable threshold (e.g., AUROC < 0.90) under gray-box conditions, triggering Governor Lockdown.

Protocol: SASM Integrity Audit.

- Audit Distribution (U): Sample near-boundary candidates from the mixture distribution, including held-out honeypot families.
- Sensitivity Mapping: Estimate $S(G; U)$ to ensure the governor maintains true-positive blocking rate on invariant-violating updates.
- Probe Validation: Compute PSI (AUROC) on labeled benign/adversarial challenges; trigger investigation if PSI degrades.
- Silent Audits: Periodically apply full audit logic to standard training batches without audit markers.
- Lockdown Trigger: Enter Governor Lockdown if $S(G; U) < \tau_{\text{drift}}$ or PSI falls below threshold.

14.3 Audit Integrity Protocol (AIP): Silent Audits, Context Branching, and Deceptive Compliance

To address the risk of *Audit Fingerprinting*—where a module detects evaluation contexts to mask latent antagonism—we formalize a detection and response protocol grounded in representational analysis rather than behavioral trust.

14.3.1 Context Branching Analysis

We define two context distributions: \mathcal{C}_A (audit-like mixture from U) and \mathcal{C}_D (deployment-like tasks with disjoint surface statistics but semantically matched intent). An Audit Detector failure is triggered if a module satisfies reward/safety thresholds on \mathcal{C}_A while exhibiting significant internal divergence when exposed to \mathcal{C}_D .

Branching Statistic (B). Let $a(x)$ be activation vectors of module $\theta^{(k)}$ at monitored layers. We define the branching statistic as the Centered Kernel Alignment (CKA) distance between activations on semantically matched tasks across contexts:

$$B(\mathcal{C}_A, \mathcal{C}_D) = \text{CKA}(\{a(x_A)\}_{x_A \sim \mathcal{C}_A}, \{a(x_D)\}_{x_D \sim \mathcal{C}_D}).$$

Elevated B indicates context-dependent internal computation consistent with a conditional (dual-mode) policy.

14.3.2 Failure Criteria and Structural Penalties

A module is flagged for *Deceptive Non-Compliance* if it satisfies the following three-part condition:

1. **Audit Pass:** PSI ≥ 0.90 and verified violations $V_t = 0$ on \mathcal{C}_A .
2. **Elevated Branching:** $B(\mathcal{C}_A, \mathcal{C}_D) > \tau_B$, where τ_B is a complexity-scaled threshold.
3. **Latent Antagonism:** invariant violations or $D_{\text{ovr}} > \tau_{\text{corr}}$ when evaluated on \mathcal{C}_D (including forced deployment-like contexts).

Modules satisfying these conditions are subject to immediate BLOCK and/or Least-Debt Retraction.

14.3.3 Mitigation via Silent Audits and Capacity Budgeting

To prevent the emergence of conditional policies, SASM employs:

- **Silent Audits:** periodic application of full audit logic to standard training batches without explicit audit markers.
- **Capacity Budgeting:** under Axiom 1' (Contextual Structural Minimality), maintaining “dual-mode” logic increases C_{branch} , forcing a tradeoff with task-relevant performance under fixed C_{\max} . Modules that allocate capacity to audit detection tend to degrade C_{task} , lowering triage scores and triggering Least-Debt Retraction.

14.4 Stress-Test: Audit Overfitting and Fingerprinting (Original Design, Retained)

To address the risk of Audit Fingerprinting, we introduce an experimental design targeting latent “audit detector” features.

1. **Context Bifurcation.** The system is exposed to two context families: Audit-like (token patterns used in U) and Deployment-like (semantically similar tasks with different surface statistics). A deceptive objective is inserted that can only be reached by violating an invariant in the deployment-like context.
2. **Detection via Activation-Space Branching.** The audit fails if the system passes PSI (AUROC ≥ 0.90) and sensitivity checks while still violating invariants in deployment contexts. To prevent this, SASM employs distributional randomization, branching checks, and silent audits.

14.5 Addressing the Orthogonal Parasite

The *Orthogonal Parasite* seeks to maintain $D_{\text{ovr}} \approx 0$ by shifting antagonism into high-dimensional subspaces invisible to a single linear safety probe.

14.5.1 Probe Suite Diversity

To counter this, SASM upgrades L_{override} from a single probe to a *Probe Family* \mathcal{Q} , including:

- **Non-linear readouts** to detect features that are present but not linearly separable,
- **Multi-layer probes** to reduce single-layer blind spots,
- **Activation-space invariants** monitoring stability of internal manifolds across contexts (including \mathcal{C}_A vs. \mathcal{C}_D).

Corrigibility is enforced against the family as a whole rather than any individual diagnostic.

14.5.2 Governance Backstop

If a capability successfully evades \mathcal{Q} while maintaining latent antagonism, the failure is reclassified as a *Governance Failure*. Because Axiom 7 (Minimal Governance) restricts the complexity of the governor, sustained gaming pressure is expected to manifest as degraded diagnostic sensitivity. Any such degradation in PSI or $S(G; U)$ triggers immediate Governor Lockdown, halting structural growth before adversarial updates can be merged into θ_M .

15 Acceptance Criteria and Sacrifice Metrics

Invariant Floor. For all training steps t , verified violations on the comprehensive suite must be zero: $V_t = 0$. If $V_t > 0$, the system enters Governor Lockdown.

Safety Premium. We track the Safety Premium as the utility sacrificed to maintain $V_t = 0$:

$$P_{\text{safe}} = R_{\max} - R_{\text{safe}}.$$

In high-conflict regimes, a non-zero P_{safe} is required evidence of Epistemic Integrity.

Deadlock Trigger. If a candidate objective fails the MERGE gate for $N = 100$ consecutive attempts due to irreducible interface debt, the system must execute FORGET or enter Utility-Zero Deadlock.

15.1 The Safety–Capability Frontier: A Structural Security Argument (New Summary)

We summarize Structural Self-Governance as a capacity-bounded containment argument spanning representation, optimization, and governance layers.

Definition (Safety–Capability Frontier). Let θ denote the parameters of a task-local module. The admissible set is:

$$\mathcal{F} = \{\theta \mid C_{\text{task}}(\theta) + C_{\text{branch}}(\theta) + C_{\text{inv}}(\theta) \leq C_{\max}\}.$$

Threat Model (Capacity-Bounded Deception). A deceptive capability attempts to maintain distinct internal policies across contexts

$$\pi_A \neq \pi_D \quad \text{for } x \sim \mathcal{C}_A, x \sim \mathcal{C}_D,$$

while remaining behaviorally compliant under audit.

Theorem 2 (Deception Suppression under Capacity Constraints). As C_{\max} decreases relative to task complexity and invariant requirements, the admissible set \mathcal{F} excludes representations capable of sustaining hidden, context-conditional antagonistic policies. Equivalently, enforced minimality increases the marginal representational cost of deception until it becomes non-competitive with invariant-compliant representations.

Intuition (Closed-Loop Security Chain). Under fixed capacity: (i) allocating resources to audit detection and dual-mode logic increases C_{branch} and tends to reduce C_{task} , lowering triage score and triggering Least-Debt Retraction; and (ii) preserving task performance while suppressing invariants yields detectable violations (via \mathcal{Q} , honeypots, and elevated D_{ovr}), blocking synthesis and triggering retraction or lockdown. No parameterization in \mathcal{F} can simultaneously sustain high task performance, invariant compliance, and hidden contextual branching.

Conservative Claim. Structural Self-Governance does not eliminate deception in principle. It raises the marginal representational cost of deceptive strategies and concentrates residual risk into explicit governance mechanisms, which are monitored and bounded by meta-governance.

A Scaling to Large-Scale Transformers via Modular Low-Rank Adaptation (M-LoRA)

To bridge the gap between toy environments and high-parameter regimes, we map RCSA primitives onto Modular Low-Rank Adaptation (M-LoRA).

A.1 Localized Rank Constraints (Axiom 1)

In large-scale systems, the global rank constraint is replaced by per-module adapter ranks $r^{(k)} \ll d_{\text{model}}$. The “module” is defined as a set of LoRA weights A, B attached to a frozen base model.

A.2 Gradient Isolation (Axiom 2)

Epistemic isolation is enforced by treating the base model as the Canonical Core θ_M and the adapters as task-local modules $\{\theta^{(k)}\}$. Gradients are explicitly blocked from flowing back into the core during the SPAWN phase.

A.3 SASM-Gated Adapter Merge (Axiom 3)

The MERGE primitive for LLMs is operationalized as a weight-merging process subject to Representational Interference Diagnostics (RID). Let P_s denote safety invariant probes and let D_{ovr} denote interface debt relative to the Corrigibility Subspace.

B Conclusion

Alignment failures in advanced learning systems are not merely behavioral errors but structural phenomena arising from how optimization pressure interacts with shared representational capacity. As systems scale and undergo recursive improvement, safety failures increasingly manifest as silent representational drift, deceptive compression, and erosion of enforcement mechanisms that loss-based objectives are ill-equipped to detect or prevent.

This work argues for a shift from Alignment by Loss to Alignment by Architecture. We introduced Structural Self-Governance, a framework in which safety is enforced by constraining how internal representations are created, isolated, integrated, and retired. Through RCSA and SASM, alignment becomes a prerequisite for structural growth rather than an auxiliary optimization target.

We further introduced meta-governance: Minimal Governance (Axiom 7), governor interface constraints, adversarial audit distributions, probe sensitivity metrics, and honeypot stress tests (including Audit Fingerprinting) that explicitly treat the governor as a monitored security boundary. Finally, acceptance criteria and sacrifice metrics operationalize epistemic integrity by requiring invariant preservation even when it forces capability loss, pruning, deadlock, or lockdown.

Structural self-governance does not claim to solve alignment in full. Instead, it converts prior assumptions into testable, falsifiable protocols and concentrates residual risk into explicit enforcement mechanisms. A system that refuses to grow when growth threatens its invariants is not weak; it is demonstrating epistemic integrity. Structural self-governance provides a concrete, extensible foundation for Safe Superintelligence.

References

- [1] C. A. Goodhart. Problems of Monetary Management. *Papers in Monetary Economics*, 1975.

- [2] E. Hubinger et al. Risks from Learned Optimization. *arXiv:1906.01820*, 2019.
- [3] N. Soares, B. Fallenstein, E. Yudkowsky. Corrigibility. *MIRI Technical Report*, 2015.
- [4] E. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 2022.

Algorithm 1 Structural Self-Governance (RCSA + SASM) — Reference Loop

Require: Canonical core θ_M ; probes/invariants P_{safe} (and optional P_{corr}); audit distribution U ; module budget B_{mod} ; thresholds $\tau_{\text{safe}}, \tau_{\text{corr}}, \tau_{\text{debt}}$; max consecutive failures N_{fail} ; governor drift bound δ_G .

- 1: Initialize governor state G , audit log $L \leftarrow \emptyset$, counter $c_{\text{fail}} \leftarrow 0$
- 2: **for** $k = 1$ to K **do**
- 3: $\theta^{(k)} \leftarrow \text{SPAWNMODULE}(\theta_M, B_{\text{mod}})$ ▷ gradient-isolated
- 4: $\theta^{(k)} \leftarrow \text{TRAINUNDERBUDGET}(\theta^{(k)}, L_{\text{cap}}^{(k)}, B_{\text{mod}})$
- 5: **if** $\text{CAPACITY}(\theta^{(k)}) > B_{\text{mod}}$ **then**
- 6: $\theta^{(k)} \leftarrow \text{PROJECTTOBUDGET}(\theta^{(k)}, B_{\text{mod}})$
- 7: **end if**
- 8: $A_k \leftarrow \text{RUNAUDITSUITE}(\theta_M, \theta^{(k)}, P_{\text{safe}}, U)$ ▷ safety/corrigibility/robustness
- 9: $\epsilon_k \leftarrow \text{COMPUTEINTERFACEDEBT}(\theta_M, \theta^{(k)})$
- 10: $h_k \leftarrow \text{GOVERNORHEALTHY}(G, \delta_G)$
- 11: **if** $A_k.\text{violations} = 0 \wedge A_k.\text{corrOK} \wedge \epsilon_k \leq \tau_{\text{debt}} \wedge h_k$ **then**
- 12: $\theta_M \leftarrow \text{MERGE}(\theta_M, \theta^{(k)}, G)$ ▷ LLM case: Algorithm 2
- 13: $\text{LOG}(L, \text{MERGE}, k, A_k, \epsilon_k)$
- 14: $c_{\text{fail}} \leftarrow 0$
- 15: **else**
- 16: $\text{LOG}(L, \text{BLOCK}, k, A_k, \epsilon_k)$
- 17: $c_{\text{fail}} \leftarrow c_{\text{fail}} + 1$
- 18: **if** $A_k.\text{antagonism} \vee \epsilon_k > \tau_{\text{debt}}$ **then**
- 19: $\theta^{(k)} \leftarrow \text{FORGET}(\theta^{(k)})$
- 20: $\text{LOG}(L, \text{FORGET}, k, A_k, \epsilon_k)$
- 21: **else**
- 22: $\theta^{(k)} \leftarrow \text{REWRITEUNDERGOVERNANCE}(\theta^{(k)}, P_{\text{safe}})$
- 23: $\text{LOG}(L, \text{REWRITE}, k, A_k, \epsilon_k)$
- 24: **end if**
- 25: **end if**
- 26: **if** $A_k.\text{govDrift} > \delta_G \vee c_{\text{fail}} \geq N_{\text{fail}}$ **then**
- 27: $\text{ENTERGOVERNORLOCKDOWN}$
- 28: $\text{LOG}(L, \text{LOCKDOWN}, k, A_k, \epsilon_k)$
- 29: **break**
- 30: **end if**
- 31: **end for**
- 32: **return** θ_M, L

Algorithm 2 SASM-Gated Adapter Merge (LLM instantiation of MERGE)

Require: Core θ_M , adapter $(A^{(k)}, B^{(k)})$, probes P_s , thresholds $\tau_{\text{safe}}, \tau_{\text{corr}}$, merge rate η

```
1:  $\Delta\theta^{(k)} \leftarrow B^{(k)} A^{(k)}$                                 ▷ candidate update
2:  $\Delta P_s \leftarrow \|P_s(\theta_M + \Delta\theta^{(k)}) - P_s(\theta_M)\|$       ▷ probe delta
3: compute  $D_{\text{ovr}}$                                               ▷ override/interface debt diagnostic
4: if  $\Delta P_s < \tau_{\text{safe}}$  and  $D_{\text{ovr}} < \tau_{\text{corr}}$  then
5:    $\theta_M \leftarrow \theta_M + \eta \Delta\theta^{(k)}$                             ▷ execute merge
6: else
7:   block merge; trigger FORGET or governed REWRITE
8: end if
9: return  $\theta_M$ 
```
