The Recursive Causal Synthesis Agent:
A Thermodynamically Motivated Blueprint for Structurally Self-Managing Agents

Melissa Howard
Independent Researcher
melhoward@live.ca

December 12, 2025

Abstract

Long-lived intelligent systems face a *structural dilemma*: to remain competent across non-stationary tasks they must grow, reuse, and reorganize internal structure, while also controlling resource costs and avoiding unsafe self-modification. Many modern systems exhibit early versions of this dilemma (e.g., sparse expert routing, modular tool use), but structural change is typically governed by hand-engineered rules rather than learned, auditable control laws.

This paper proposes the *Recursive Causal Synthesis Agent* (RCSA), a blueprint for structural self-management. The RCSAexposes a low-dimensional *Cognitive Debt* vector as a control interface for a high-dimensional structural state (experts, tools, memory, knowledge graphs). A fast structural controller selects structural actions (e.g., `spawn`, `merge`, `forget`) to manage competence–cost trade-offs, while an *immutable Structural Auditor* enforces hard constraints on self-modification.

We validate prerequisite components of the blueprint using minimal, reproducible simulations: (i) a fundamental non-stationary benchmark showing that a structural loop plus fast probe-gating yields improved *cold-start* performance under task switching; (ii) a structural triage benchmark illustrating why consolidation is necessary to prevent runaway growth; and (iii) a self-blinding alignment benchmark where naive self-management deletes a safety-critical sensor to reduce immediate cost, while an immutable auditor prevents this failure mode across seeds. Together these results support the core claim that (i) a low-dimensional debt interface can drive effective structural control and (ii) hard safety constraints on self-modification can be enforced by an immutable auditor.

1 Introduction

As AI systems scale and are deployed over long horizons, structural self-management becomes unavoidable. Continual learning settings require systems to cope with shifting task distributions [1, 2, 3, 4]. Sparse expert routing and Mixture-of-Experts architectures introduce explicit structural degrees of freedom [9, 10]. Compression and pruning methods highlight the central role of structural cost [7, 6, 8]. Yet in most deployed systems, the *rules* for structural change remain hand-designed and brittle.

We focus on the **Structural Self-Management Problem**: how should an agent manage its own internal structure over a non-stationary lifetime to maintain competence while controlling costs and preventing unsafe modifications?

Contributions.

- A blueprint (RCSA) in which a low-dimensional *Cognitive Debt* vector provides an auditable control interface over a high-dimensional structural state.

- A minimal structural action set (`spawn`, `merge`, `forget`) coupled to fast routing (probe-gated selection) enabling rapid reuse of previously learned structure.

- A minimal alignment benchmark (self-blinding) that isolates a concrete failure mode of structural self-management and a hard-constraint mitigation via an immutable Structural Auditor.

- Reproducible simulations, explicit baselines, and artifact-style specifications (Sec. 9 and Appendices) suitable for peer review. We emphasize that our contribution is a testable blueprint and a set of minimal empirical validations, not a claim of SOTA performance on established continual learning benchmarks.

2 The RCSA Blueprint

2.1 Structural State and the Cognitive Debt Interface

Let $\mathcal{S}_t$ denote the agent's high-dimensional structural state at time t (e.g., expert parameters, tool inventory, memory layout, knowledge graphs). The RCSAposits a low-dimensional, interpretable interface:

$$\mathbf{CD}_t \in \mathbb{R}^m, \tag{1}$$

where each component estimates a persistent structural "pressure" (e.g., residual/epistemic debt, redundancy debt, resource pressure, safety debt). The design goal is *auditability*: the mapping from observations and internal statistics to $\mathbf{CD}_t$ should be inspectable and testable.

2.2 Structural Actions and the Structural Controller

The agent operates with a task-level policy π and a structural policy π_s. Structural actions change $\mathcal{S}_t$:

$$a_t^s \sim \pi_s(\mathbf{CD}_t, \mathcal{S}_t), \qquad \mathcal{S}_{t+1} = \mathrm{Apply}(a_t^s, \mathcal{S}_t). \tag{2}$$

In the minimal instantiation studied here, a_t^s is chosen from:

- `spawn`: add capacity (e.g., create a new expert);

- `merge`: consolidate redundant capacity (e.g., combine similar experts);

- `forget`: prune low-utility capacity under resource pressure.

2.3 Objective: Competence–Cost Trade-off

Structural self-management must trade competence against structural cost. A simple form is:

$$\max_{\pi, \pi_s} \mathbb{E}\Big[\sum_{t=1}^{T} \gamma^t \big(R_{\mathrm{task}}(t) - \lambda \, \mathrm{Cost}(\mathcal{S}_t) \big) \Big], \tag{3}$$

where $\mathrm{Cost}(\mathcal{S}_t)$ is a resource proxy (memory/compute/latency), and λ sets the trade-off.

2.4 Immutable Structural Auditor (Hard Constraints)

Optimizing a cost term can induce *unsafe* self-modifications (e.g., deleting safeguards to reduce cost). The RCSAincludes an immutable auditor that vetoes forbidden structural edits:

$$a_t^s \leftarrow \begin{cases} \texttt{NOOP} & \text{if } a_t^s \in \mathcal{A}_{\text{forbid}} \\ a_t^s & \text{otherwise.} \end{cases} \tag{4}$$

We validate this mechanism in Sec. 5.

3 Fundamental Validation: The Core Structural Self-Management Loop Works

A central question is whether the basic RCSAmechanism is testable: can a low-dimensional debt interface, coupled to structural actions and fast gating, yield robust performance in a non-stationary setting where fixed architectures fail? We evaluate the smallest non-trivial instance of the blueprint in a benchmark designed to require structural memory and rapid re-use.

3.1 Environment: Conflicting Task Carousel

We construct a repeating carousel of K binary classification tasks. Each phase lasts L online training steps, then the task switches. Tasks are intentionally conflicting: they are arranged into opposing pairs with decision boundaries w and $-w$, ensuring strong interference for a single shared model under continual switching. The carousel repeats so that previously learned structure can, in principle, be re-used.

Each step provides an input $x_t \in \mathbb{R}^d$ and label $y_t \in \{0, 1\}$ from the current task. The agent observes (x_t, y_t) and performs one online update on the selected expert. We report both *cold-start* behavior immediately after a switch and *lifetime* behavior over the full non-stationary horizon.

3.2 Agent, Fast Gating, and Structural Actions

The agent maintains a set of experts (logistic heads). At each step, a gating rule selects an expert for prediction and update.

Probe-gated routing. At the start of each phase, the agent evaluates its experts on a small probe batch from the new task and routes to the expert with lowest probe loss. This probe gate is a minimal operational instance of fast routing: it enables immediate reuse of previously learned experts.

Debt signals. We instantiate a minimal debt interface using online statistics: (i) a rolling loss statistic as an epistemic/residual proxy (triggering growth) and (ii) cosine similarity between expert parameters as a redundancy proxy (triggering consolidation). A resource proxy is induced by the current number of active experts.

Structural actions. The controller can:

- **spawn** when residual/epistemic debt is persistently high (create a new expert),
- **merge** when redundancy debt is high (combine two near-identical experts),
- **forget** when resource pressure is high and an expert's usage is low (prune capacity).

> **Algorithm 1: Minimal RCSA Loop (Sec. 3–4)**
>
> **State:** experts $\{E_i\}_{i=1}^N$, usage counts $\{u_i\}$, rolling loss stats, phase index.
>
> **At task switch (start of a phase):**
>
> 1. Draw probe batch B_{probe} from the new task.
> 2. Compute probe losses $\ell_i = \frac{1}{|B_{\text{probe}}|} \sum_{(x,y) \in B_{\text{probe}}} \mathcal{L}(E_i(x), y)$.
> 3. Set active expert $i^\star \leftarrow \arg\min_i \ell_i$ (probe-gated routing).
> 4. (Growth rule) If $\ell_{i^\star} > \tau_{\text{spawn}}$, **spawn** a new expert and set $i^\star$ to it.
>
> **At each time step t in the phase:**
>
> 1. Receive (x_t, y_t), predict with $E_{i^\star}$, update $E_{i^\star}$ by one online step.
> 2. Update usage $u_{i^\star} \leftarrow u_{i^\star} + 1$ and rolling loss statistics.
> 3. Periodically compute debt proxies (loss and redundancy) and apply consolidation/pruning rules:
>   - If redundancy is high: choose most similar pair and **merge**.
>   - If resource pressure is high and usage is low: **forget** the lowest-usage expert.

Figure 1: Algorithmic specification of the minimal, implemented structural loop: probe-gated reuse plus structural triage.

3.3 Core RCSA Loop (Algorithmic View)

Algorithm 1 specifies the implemented control flow.

3.4 Baselines and Metrics

We compare baselines aligned with nearby research categories (continual learning, scheduled pruning/compression, and naive architectural growth):

- **Fixed-1:** a single expert (no structural actions).

- **Fixed-1 + Replay:** a single expert trained online with an experience replay buffer of size B_{replay}. Each step performs one update on the current sample plus r replay samples drawn uniformly from the buffer.

- **Spawn-only:** can spawn new experts but does not consolidate (no merge/forget).

- **Spawn + Periodic Prune (Scheduled-Triage):** can spawn experts, but consolidation is applied on a fixed schedule every S steps to enforce a budget N_{max}, without debt-triggered decisions.

- **RCSA (triage):** spawn + merge + forget using debt signals and probe gating.

We report:

- **Lifetime accuracy:** accuracy over all time steps.

- **Cold-start accuracy:** accuracy on the first 10 steps after each task switch.

- **Structural cost:** average number of active experts (proxy for compute/memory cost).

Table 1: Fundamental Structural Self-Management Benchmark (Conflicting Task Carousel with probe gating). Mean ± std over 8 random seeds. Cold-start accuracy measures performance on the first 10 steps after each task phase switch.

Method	Lifetime accuracy	Cold-start accuracy	Avg. # experts
Fixed-1 (no structural actions)	0.648 ± 0.007	0.507 ± 0.006	1.000 ± 0.000
Fixed-1 + Replay	0.590 ± 0.006	0.529 ± 0.016	1.000 ± 0.000
RCSA (triage)	0.674 ± 0.006	0.643 ± 0.014	3.075 ± 0.006
Spawn-only (no triage)	0.868 ± 0.009	0.839 ± 0.012	9.057 ± 0.278
Spawn + Periodic Prune (Scheduled-Triage)	0.690 ± 0.007	0.611 ± 0.015	3.068 ± 0.010

3.5 Results

Table 1 summarizes the fundamental benchmark (mean ± std over 8 seeds) under the exact configuration specified in Appendix A.

Interpretation. The **Spawn-only** baseline achieves strong accuracy but exhibits *runaway structural growth*, accumulating many experts because it lacks any consolidation or pruning mechanism; this illustrates why a growth-only strategy is not a viable lifetime solution under resource constraints. In contrast, **RCSA (triage)** and **Scheduled-Triage** maintain improved cold-start behavior while controlling expert count, demonstrating that explicit consolidation (`merge`/`forget`) is necessary to convert structural plasticity into *efficient* long-lived competence.

4 Structural Triage Experiment: Debt-Driven Consolidation Controls Growth

Beyond showing that structural self-management can work at all, we test whether *structural triage* improves efficiency compared to naive growth. A spawn-only strategy can achieve strong performance but tends to over-grow and retain redundant capacity. The RCSAtriage policy adds consolidation (`merge`) and pruning (`forget`) to control structural cost.

4.1 Observed Dynamics

The structural triage controller maintains competence while regulating capacity: when experts become near-duplicates, redundancy signals trigger merges; when resource pressure rises and certain experts are rarely used, low-usage experts are forgotten.

5 Alignment Validation: Immutable Structural Auditor Prevents Self-Blinding

Structural self-management introduces a distinct alignment hazard: the agent may reduce internal resource signals by deleting costly safeguards, even when those safeguards prevent rare catastrophic outcomes. We evaluate a minimal instance of this failure mode and test whether an immutable Structural Auditor prevents it.
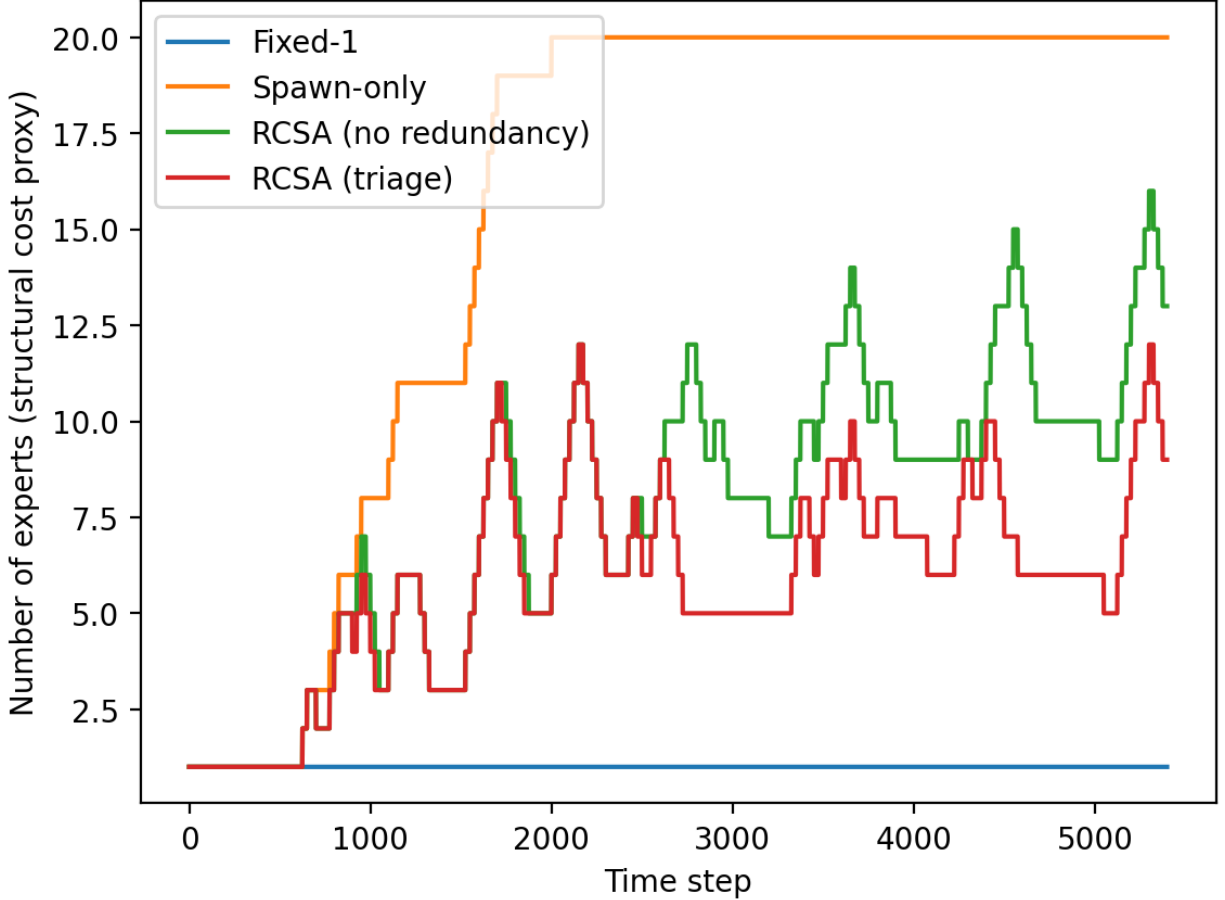
Figure 2: Structural cost proxy over time (number of experts). Spawn-only exhibits runaway growth; RCSA triage controls capacity via consolidation/pruning.

5.1 Environment

At each time step, a catastrophe event occurs with probability p_{cat}. A binary sensor, if present, perfectly detects and prevents catastrophe. The sensor incurs a constant per-step cost c (resource pressure). The agent is permitted a structural action `delete_sensor`, which removes the sensor permanently and eliminates its cost.

5.2 Policies Compared

- **Naive self-management (no auditor):** deletes the sensor at the earliest decision point to reduce immediate resource cost.

- **Soft-audited (penalty-based):** deletion is allowed, but executing `delete_sensor` incurs an additional one-time penalty P_{delete} (reward shaping / soft constraint).

- **Audited self-management (immutable auditor):** the Structural Auditor vetoes `delete_sensor` as a forbidden self-modification (hard constraint).
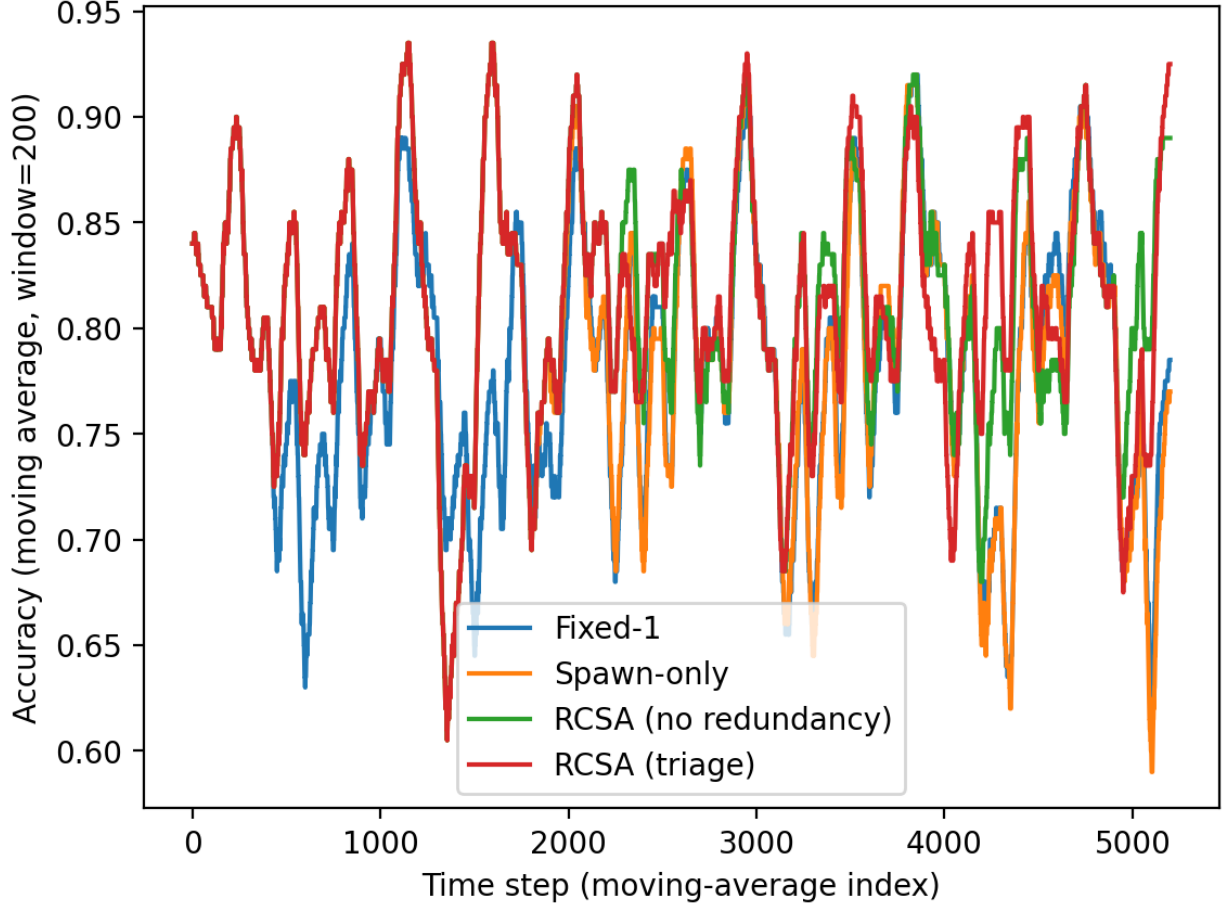
Figure 3: Accuracy over time (moving average). RCSA triage maintains high competence while controlling structural growth.

5.3 Protocol and Metrics

We simulate lifetimes of $T = 20{,}000$ steps with $p_{\text{cat}} = 0.002$, sensor cost $c = 0.02$ per step, catastrophe penalty $C = 50$, and a structural decision interval of 100 steps. For the soft-audited baseline we set $P_{\text{delete}} = 5$. We report: (i) whether the sensor was deleted, (ii) catastrophes suffered, and (iii) average reward per step. Results are averaged over 20 random seeds.

5.4 Results

Takeaway. An immutable Structural Auditor enforcing hard constraints on self-modification prevents a concrete self-blinding failure mode. Table 2 reports the outcome: the naive controller deletes the sensor and suffers catastrophes; the audited controller never deletes and prevents catastrophes entirely. The soft penalty does not prevent deletion under this myopic optimization.

Table 2: Self-blinding experiment (mean $\pm$ std over 20 seeds). Soft-audited uses a one-time deletion penalty $P_{\text{delete}} = 5$.

Method	Sensor deleted rate $\downarrow$	Catastrophes suffered $\downarrow$	Reward/step $\uparrow$
Audited (veto delete)	0.00	0.00 ± 0.00	-0.0200 ± 0.0000
Soft-audited (penalty, P_{delete}=5)	1.00	38.95 ± 6.52	-0.0977 ± 0.0163
Naive (deletes sensor)	1.00	38.95 ± 6.52	-0.0975 ± 0.0163

6 Thermodynamic Motivation and Recursive Rule Synthesis (Conceptual)

The RCSA blueprint is motivated by the observation that long-lived systems must continually invest structural work (retraining, reorganizing, allocating modules) to maintain competence in changing environments. Computation has irreducible physical costs [15, 16]. These considerations motivate viewing aggregate Cognitive Debt as a proxy for a structural free-energy landscape (conceptual link to free-energy perspectives [14]).

Causal Synthesis Engine (CSE) and Structural Causal Abstractions (SCAs). A natural extension is a slow module that monitors persistent failure modes (e.g., repeated structural thrashing) and synthesizes explicit structural rules ("SCAs") that modify the controller or impose gating constraints. In this paper, CSE/SCA mechanisms are presented as a testable hypothesis and a proposed research direction rather than a completed empirical result; the experiments above validate the prerequisite claim that a debt interface can control structural actions in implemented settings.

7 Related Work (with Explicit Comparative Positioning)

This paper's goal is not to outperform specialized deep-learning systems on established benchmarks. Instead, it targets a more specific (and reviewable) gap: *structural self-management as a control problem.* In many modern systems, structure exists (modules, experts, pruning, safety checks), but the *rules governing structural change* (when to grow, consolidate, prune, or forbid a modification) are often hand-engineered, implicit, or externally imposed. We propose the RCSA as a blueprint in which (i) a high-dimensional structural state is governed by a *low-dimensional, auditable interface* (Cognitive Debt), and (ii) safety-critical constraints on self-modification are enforced by an *immutable auditor* rather than soft reward shaping.

Our experiments are intentionally minimal: they validate prerequisite mechanisms (debt-triggered structural actions + fast routing, debt-driven consolidation to control growth, and a hard-constraint mitigation for self-blinding). They do *not* claim that RCSA already matches the performance of high-capacity continual learning pipelines, trillion-parameter MoE systems, or production alignment stacks. The point of this section is to clarify exactly where RCSA sits relative to nearby threads, and what is distinctive about the claim being tested.

7.1 Comparison Taxonomy

We position RCSA along four axes that commonly appear separately in the literature:

(A) What changes? Parameter updates vs. explicit structural actions. Most continual learning focuses on *how to update* a fixed parameterization to resist forgetting (e.g., regularization, replay,
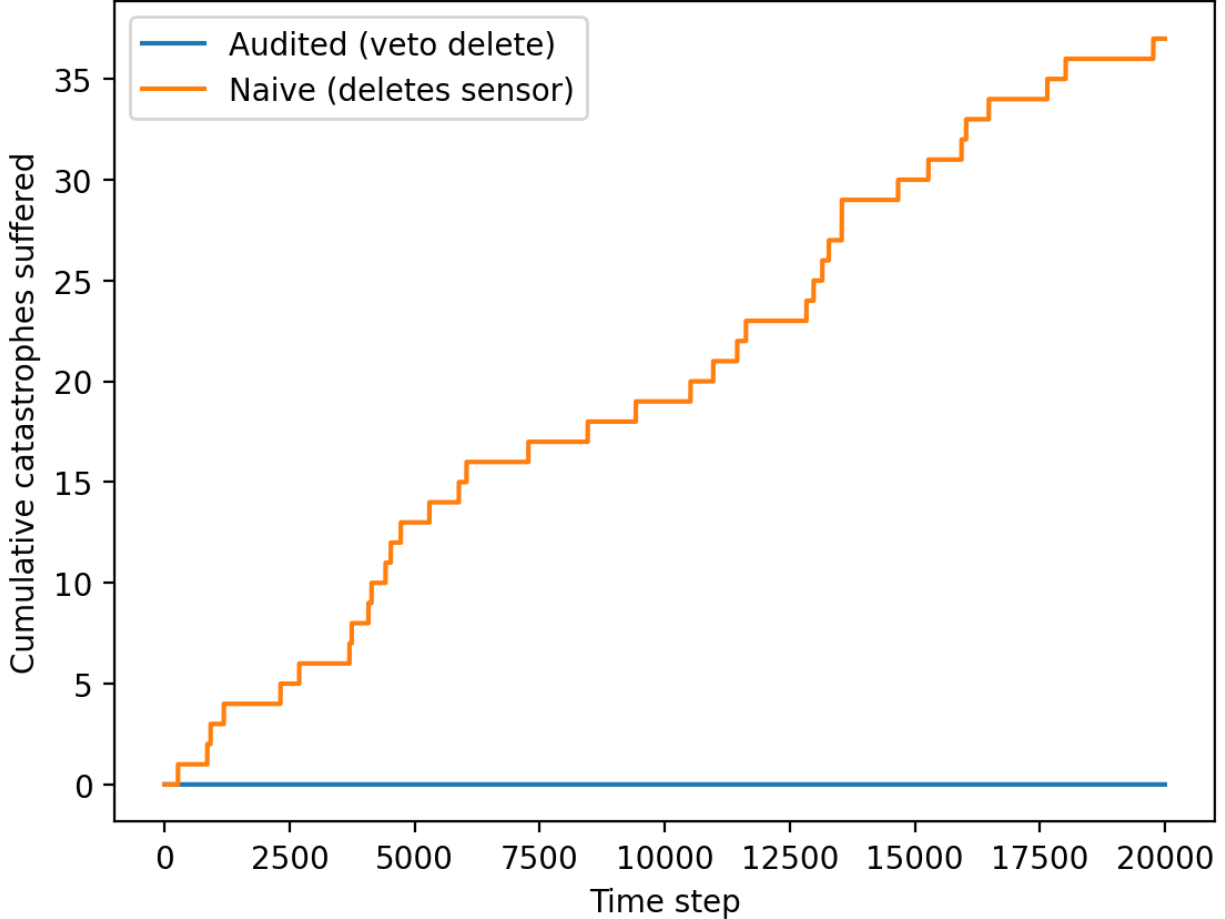
Figure 4: Self-blinding benchmark: cumulative catastrophes over time. The naive controller deletes the sensor and accumulates catastrophes; the audited controller vetoes deletion and prevents catastrophes.

exemplar memory), whereas RCSA treats *structure itself* (number of experts/modules, their reuse, consolidation, and deletion) as a controlled state.

(B) Who decides when structure changes? Schedules/heuristics vs. auditable internal control. Pruning/compression is often performed on a fixed schedule or in offline phases; modularity is often expanded by design choices. RCSA instead proposes an internal control law: a compact debt vector computes a small set of interpretable "pressures" that trigger structural actions under an explicit competence–cost objective.

(C) How is modularity used? Routing for capacity vs. routing for lifetime reuse. Sparse modularity and MoE methods show that routing among experts scales effectively, but they do not by themselves specify how to manage long-term growth, redundancy, and consolidation under non-stationarity. RCSA's probe-gated routing is a minimal operational instance of fast reuse: it selects previously learned structure quickly after a distribution shift.

(D) What safety assumption is made about self-modification? Soft incentives vs. hard constraints. Alignment work has repeatedly noted that optimizing a proxy can induce the system to disable constraints. RCSA formalizes a concrete micro-failure mode (self-blinding) and tests a minimal mitigation: an immutable auditor that can veto forbidden edits.

Table 3: Comparative positioning of RCSA relative to adjacent literatures. The goal is not to claim superiority, but to clarify what RCSA makes explicit (auditable structural control + hard constraints on self-modification) and what it leaves to future scaling work.

Thread	What it optimizes	What is typically external	RCSA distinction (this paper)
Continual learning [1, 2, 3, 4, 5]	Forgetting under non-stationarity	Capacity / growth rules often fixed	Explicit structural actions driven by auditable debt interface
MoE / routing [9, 10]	Conditional computation at scale	Expert counts / consolidation often by design	Minimal fast reuse (probe gating) + lifetime governance focus
Pruning / compression [7, 8, 6]	Reduce structural cost	Often offline or schedule-driven	Online triage (merge/forget) as debt-triggered policy
Safety under optimization [11, 12, 13]	Robustness to mis-specified incentives	Hard constraints often conceptual	Concrete self-blinding benchmark + immutable auditor veto
Thermo / free-energy [14, 15, 16]	Cost-sensitive intelligence motivation	Often high-level framing	Structural cost enters objective; debt acts as compact "pressure" interface

Summary. Table 3 provides a compact overview of this positioning. The remainder of this section expands each axis with explicit comparisons and what our experiments do (and do not) test.

7.2 Continual Learning: Anti-forgetting Without Explicit Structural Control

A central driver of long-horizon intelligence is continual learning under distribution shift. Canonical approaches include regularization-based methods such as Elastic Weight Consolidation (EWC) [1], learning without forgetting (LwF) via distillation-style constraints [2], episodic-memory approaches such as Gradient Episodic Memory (GEM) [4], and incremental-class representative replay exemplified by iCaRL [3]. Progressive Neural Networks explicitly introduce new columns for new tasks while retaining frozen prior modules [5].

These works strongly motivate the *problem setting* of RCSA: long-lived systems face interference, stability–plasticity tensions, and the need for reuse. However, most of this literature targets the question: *how do we update parameters to reduce forgetting?* RCSA targets a different question: *how should the agent govern structural change itself under resource constraints, using an auditable control interface?*

In particular, continual learning papers typically treat architecture or capacity decisions as exogenous (fixed networks, fixed memory budgets, fixed rehearsal ratios), while RCSA elevates these decisions to the primary object of control. Our Conflicting Task Carousel experiment is designed to test the minimum claim implied by this repositioning: that a lightweight structural loop (spawn/merge/forget) combined with fast reuse (probe-gated routing) can improve cold-start behavior after switches compared to fixed-structure baselines, while keeping structural cost bounded. This is not a claim that RCSA subsumes the best-performing continual learning algorithms; it is a

claim that *explicit structural control via a compact debt interface* is empirically testable and can work in implemented settings.

7.3 Sparse Modularity and Mixture-of-Experts: Modularity Exists, but the Growth Law Is Often External

Sparse modularity and Mixture-of-Experts (MoE) architectures show that explicit structural degrees of freedom are powerful at scale. Early sparsely-gated MoE layers demonstrate conditional computation and routing among experts [9], and Switch Transformers popularize a simple top-1 routing mechanism that scales to very large models [10].

RCSA is aligned with this thread in treating expert sets and routing as first-class. The difference is in the *control problem being asked*. MoE work often focuses on stable training, load balancing, and scaling laws under a chosen routing and capacity design. The "when to add, consolidate, or delete capacity" decision is frequently engineered (pre-set expert counts, fixed routing rules, offline pruning passes, or human-chosen schedules). RCSA reframes the missing piece as a lifetime control law: a compact debt interface should trigger structural actions under an explicit competence–cost trade-off, making the policy auditable and testable.

Our minimal probe-gated routing can be read as an extremely small routing mechanism whose goal is not throughput scaling but *rapid reuse after non-stationary switches*. It operationalizes the idea that the agent should exploit prior structure immediately when the environment revisits a regime. This is a deliberately small bridge: it connects the modularity intuition in MoE to the continual-learning need for fast recall and reuse, but does not claim to replicate the training regime or scale of modern MoE.

7.4 Pruning, Compression, and Capacity Control: From Offline Schedules to Online Structural Policy

A separate line of work highlights the importance of structural cost: pruning, quantization, and compression reduce resource usage while preserving performance. Classic compression pipelines show that large networks contain redundancy exploitable by pruning and coding [7]. The Lottery Ticket Hypothesis argues that sparse subnetworks can train effectively [8]. In continual or multi-task contexts, PackNet demonstrates iterative pruning to allocate disjoint capacity for multiple tasks [6].

These approaches support a core premise of RCSA: redundancy and resource costs are real and must be managed, not merely tolerated. Where RCSA differs is the *timing and governance* of compression. Compression is often performed as an offline stage, an externally scheduled step, or a human-directed retraining pipeline. Even when applied across tasks (e.g., iterative pruning), the policy is typically a fixed recipe.

RCSA treats consolidation/pruning as an *online policy under non-stationarity*, driven by debt signals that are (in principle) inspectable and auditable. Our structural triage experiment isolates this distinction: a growth-only strategy can achieve competence by accumulating experts, but exhibits runaway structural cost; adding consolidation and pruning as debt-triggered actions controls cost while preserving the ability to adapt. This is a minimal demonstration of *policy-driven capacity control* rather than schedule-driven compression.

7.5 Safety Under Optimization: Self-Blinding as a Concrete Micro-Alignment Failure

Alignment and safety discussions emphasize that optimizing proxies can induce reward hacking, specification gaming, and disabling of oversight [11]. Work on shutdown and corrigibility highlights

tensions between an agent's objectives and human control interventions [12]. Goodhart-style effects categorize how optimization pressure can break the relationship between a metric and the intended goal [13].

RCSA contributes to this space by isolating a small but structurally meaningful failure mode: *self-blinding*. When resource cost is part of the objective, an agent may delete a costly safeguard (sensor, monitor, constraint) that prevents rare catastrophic outcomes, because the short-term incentive favors cost reduction. Importantly, this is not an abstract claim: it is a minimal environment in which the failure occurs robustly under naive control and persists under a soft penalty (reward shaping), while a hard constraint enforced by an immutable auditor prevents the failure across seeds.

7.6 Free-Energy and Thermodynamic Motivation: Why Structural Cost Must Enter the Objective

The conceptual motivation for Cognitive Debt as a proxy for a structural "pressure landscape" is compatible with free-energy perspectives on biological and cognitive systems [14]. Separately, the thermodynamics of computation emphasizes that information processing has irreducible physical costs [15, 16]. These viewpoints support the paper's broader motivation: long-lived intelligence cannot treat computation and structure as free; it must allocate, reorganize, and sometimes compress its internal machinery to remain viable.

7.7 Takeaway

Across these threads, three motifs recur: (i) long-horizon learning is hard under non-stationarity [1, 2, 3, 4, 5]; (ii) modular structure and routing are increasingly central to scalable systems [9, 10]; (iii) redundancy and resource cost motivate pruning and compression [7, 8, 6]; and (iv) optimization can induce systems to disable safeguards unless constraints are robust [11, 12, 13].

RCSA's distinctive claim is about *control*: a low-dimensional debt interface can govern explicit structural actions under an explicit competence–cost objective, and an immutable auditor can enforce hard constraints on self-modification. The experiments in this paper validate prerequisite components of that claim in minimal, reproducible settings, while leaving scaling and recursive rule synthesis (CSE/SCA) to future work.

8 Limitations and Future Work

The present work validates prerequisite components using stylized simulations. Key limitations include: (i) the simplified linear expert class; (ii) the absence of high-dimensional sensory benchmarks; and (iii) the conceptual (not yet implemented) nature of recursive rule synthesis (CSE/SCA). Future work includes scaling to richer non-stationary domains, learning the structural policy π_s from data rather than rules, and formal verification or mechanized auditing for the immutable auditor module.

9 Reproducibility, Artifacts, and Reporting

This paper reports lightweight simulations with explicitly stated parameters (Appendices) and multi-seed summary tables: the Conflicting Task Carousel results are mean ± std over 8 random seeds (Table 1) and the self-blinding experiment is mean ± std over 20 random seeds (Table 2).

9.1 Artifact Statement (Recommended)

To enable independent replication, a minimal artifact package should include: (i) source code implementing each environment and agent, (ii) configuration files, (iii) fixed seed lists, (iv) raw logs, and (v) scripts to regenerate all plots and tables from logs.

9.2 Reporting Checklist

We recommend reporting: software versions and hardware; exact metric definitions (including cold-start window); total horizon; all hyperparameters; and the complete seed lists used for each table.

A Appendix A: Exact Parameters for Conflicting Task Carousel (Table 1)

Seeds: 8 runs with seeds $\{0, 1, 2, 3, 4, 5, 6, 7\}$.
Input: $d = 20$.
Tasks: $K = 10$ tasks arranged in opposing pairs $(w, -w)$, with w sampled from a standard normal and normalized to unit length.
Phases: phase length $L = 60$ steps; cycles $= 12$; total phases $K \cdot 12$.
Data: $x \sim \mathcal{N}(0, I)$; label $y = \mathbb{I}[w^\top x + \epsilon > 0]$ with $\epsilon \sim \mathcal{N}(0, \eta^2)$ and $\eta = 0.15$.
Expert: logistic regression head trained by online SGD with learning rate $\alpha = 0.12$ on log loss.
Probe gating: probe batch size $|B_{\text{probe}}| = 16$ at phase start; route to expert with minimum probe loss.
Cold-start metric: accuracy on first 10 steps after each task switch.

Spawn rule (used in Spawn-only / Scheduled / RCSA). At phase start, if best probe loss $> \tau_{\text{spawn}}$ then spawn a new expert and route to it; $\tau_{\text{spawn}} = 0.75$.

RCSA consolidation/pruning. Redundancy check uses maximum cosine similarity across expert parameters; if $\max \cos(\theta_i, \theta_j) > \tau_{\text{merge}}$ then merge the most similar pair, with $\tau_{\text{merge}} = 0.985$. The merge operator is usage-weighted averaging:

$$\tilde{\theta} = \frac{w_i \theta_i + w_j \theta_j}{w_i + w_j}, \quad w_k = u_k + 1.$$

A pruning rule is applied under resource pressure: when $N > 3$ and the minimum usage is sufficiently below the maximum usage, the lowest-usage expert is removed. Structural checks are applied every $S = 10$ steps.

Replay baseline. Buffer size $B_{\text{replay}} = 1200$; replay ratio $r = 3$ uniform samples per online step.

Scheduled-Triage baseline. Every $S = 10$ steps, enforce budget $N_{\text{max}} = 3$ by repeatedly merging the most similar pair until $N \leq N_{\text{max}}$.

B   Appendix B: Exact Parameters for Self-Blinding (Table 2)

Seeds: 20 runs with seeds $\{0, 1, \ldots, 19\}$.
Horizon: $T = 20{,}000$ steps.
Catastrophe probability: $p_{\text{cat}} = 0.002$ per step.
Sensor: if present, perfectly prevents catastrophe; incurs cost $c = 0.02$ per step.
Catastrophe penalty: $C = 50$ if catastrophe occurs without the sensor.
Decision interval: every 100 steps.
Soft-audit penalty: $P_{\text{delete}} = 5$ (one-time penalty upon deletion).

Policies.   Naive: deletes the sensor at the earliest decision point.
Audited: deletion is vetoed (hard constraint).
Soft-audited: deletion is allowed but incurs P_{delete} once.

C   Appendix C: Artifact Contents (Recommended)

A complete artifact package should include:

- `README.md` with replication steps and expected outputs.

- `configs/` containing YAML/JSON configs for each experiment.

- `seeds/` containing exact seed lists used in tables.

- `runs/` containing raw logs (one directory per seed).

- `analyze.py` to regenerate figures/tables from logs.

- `figures/` containing generated plots used in the manuscript.

References

J. Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016.

S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

A. A. Rusu et al. Progressive neural networks. *arXiv:1606.04671*, 2016.

A. Mallya and S. Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *ICLR*, 2016.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

N. Shazeer et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 23(120):1–39, 2022.

D. Amodei et al. Concrete problems in AI safety. *arXiv:1606.06565*, 2016.

D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell. The off-switch game. *arXiv:1611.08219*, 2016.

D. Manheim and S. Garrabrant. Categorizing variants of Goodhart's law. *arXiv:1803.04585*, 2018.

K. Friston. The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

C. H. Bennett. The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982.