

Stateful Adaptive Systems: Attractor Dynamics, Recovery, and Meta-Stability

Chen Yao Sheng

Independent Researcher

elroy.chen@proton.me

ORCID: <https://orcid.org/0009-0004-6984-2326>

Preprint. Under review.

Abstract

We present a cognitive runtime architecture that decouples language generation from safety-critical decision-making. A 384-dimensional sentence embedding is projected through a learned linear map into a 32-dimensional continuous cognitive field. A structured attractor graph biases this projection, implementing top-down attention: active memory reshapes how semantic input is perceived. The field state is mapped to a safety verdict without routing through the language model, which serves exclusively as an I/O layer. All components learn online from binary outcome feedback via Hebbian updates.

A four-turn deployment scenario validates that the architecture produces adaptive caution—defaulting to vigilance, resisting user pressure, and relaxing as operational evidence accumulates. However, a 50-turn longitudinal experiment reveals a structural pathology: under asymmetric outcome feedback, the system collapses into a BLOCK-dominated absorbing state (48% overall, 100% sustained from turn 36). Four mechanisms—weight decay, non-linear world override, Oja’s Rule, and outcome credit assignment—collectively reduce the BLOCK rate to 0% in the final configuration.

A controlled AB comparison quantifies the difference between stateless and stateful cognition: the runtime produces strong temporal autocorrelation ($r = 0.938$) versus near-random behavior ($r = 0.052$) in the stateless baseline. We define a meta-stability framework with five sub-measures; a revised context-conditional formulation achieves $151.7\times$ mean-ratio separation ($57.5\times$ worst-case) between healthy and pathological trajectories, with robustness confirmed by a 14-variant sensitivity sweep.

The system runs on consumer hardware with no training corpus. All code is open-source.

Keywords: cognitive architecture, attractor networks, Hebbian learning, stateful agents, meta-stability, top-down attention

1. Introduction

1.1 Motivation: The Failure of Context-Space Learning

This paper is the fourth in a series that began with an investigation of how language models interpret injected behavioral priors (Chen, 2026b) and proceeded through simulation and runtime evaluation of an automated distillation architecture (Chen, 2026c, 2026a).

A prior study found that LLMs extract *functional intent* from prior definitions and re-implement it in context, independent of keyword surface semantics (Chen, 2026b). This finding motivated the ADAM 2.0 architecture: a five-layer memory pipeline (L0→L3) that distills conversational experience into structured cognitive priors, which are injected into subsequent session prompts to steer behavior.

A controlled simulation (Chen, 2026c) predicted that this L0→L3 distillation loop would produce compounding behavioral improvement—+88% compliance lift over no-prior baseline. The simulation assumed static prior injection and could not model the dynamics of a growing prior store.

A runtime evaluation (Chen, 2026a) tested this prediction in the full ADAM 2.0 runtime over 30 structured operational sessions. The result contradicted the simulation. As L3 cognitive priors accumulated from 0 to 45, behavioral compliance degraded from a first-five-session mean of $S=0.56$ to a last-five-session mean of $S=0.36$ —a learning delta of $\Delta S=-0.200$. The system got worse as it learned more.

We identified the cause as **context-space entropy growth**: as priors accumulate, each injected into a finite context window competes with all others for the model’s attention. The signal-to-noise ratio per prior decreases monotonically with store size. This is not a retrieval algorithm failure—it is a structural property of injecting text into a stateless model. A prior stored in L3 and injected at session start is re-presented to a transformer that has no persistent latent state between calls. The system accumulates text; the model accumulates nothing.

The diagnosis led to a specific architectural conclusion: **a transformer is not a cognition substrate**. It is well-suited as a semantic interface between human language and structured internal representations. It is not suited for persistent behavioral adaptation across sessions, because it cannot accumulate experience—it can only receive text that describes past experience.

1.2 This Paper: Moving Cognition Out of the LLM

ADAM 3.0 is the architectural response to the diagnosis in Chen (2026a). We remove the LLM from the decision pathway entirely. It does not reason, plan, assess risk, or select actions. It receives a structured verdict from a separate cognitive runtime and renders it in natural language.

The runtime is small (a 32×384 projection matrix, a 32-dimensional field, a $32 \rightarrow 1$ safety head), fully observable (every dimension can be logged and visualized), and continuously learning (all weights update from binary outcome feedback). It accumulates experience not as text in a context window, but as weights in a learned projection matrix, coupling strengths in a dynamical field, and connection strengths in an attractor graph—all of which persist across sessions and improve with accumulation rather than degrading.

1.3 The Problem with LLM-as-Brain (General Case)

Beyond the specific failure documented in Chen (2026a), the LLM-as-brain paradigm creates three structural problems that affect any agent architecture where the language model is the decision-maker. **Opacity**: tracing a verdict requires interpreting token probabilities across billions of parameters; there is no intermediate cognitive state to inspect. **Fragility**: safety depends on training-internalized patterns plus prompt engineering—both vulnerable to prompt injection, jailbreaks, and distribution shift, with the decision logic and the vulnerability surface being the same component. **Entanglement**: when the LLM is both safety assessor and response generator, helpfulness and safety objectives share the same parameter space and trade off against each other.

1.4 Memory Biases Perception: The Key Insight

The key architectural insight is that memory should bias perception. In our system, cognitive attractors—structured memory units organized as a graph with learned connections—do not merely compete for activation after semantic input is projected into cognitive space. They bias the projection itself. When safety-related attractors are active, the same user utterance projects to a different region of the 32-dimensional cognitive field than when execution-oriented attractors dominate. The word “deploy” lands in a different cognitive location depending on what the system remembers about deployment.

This is analogous to biased competition in biological visual attention (Desimone & Duncan, 1995), where top-down signals from prefrontal cortex modulate sensory processing in visual areas before conscious perception occurs. Our attractor bias implements a learned, continuous form of this modulation—not in visual cortex, but in the projection from semantic embedding space to cognitive decision space.

1.4a Relevance to Developmental Cognitive Systems

While this paper’s motivation is the failure documented in Chen (2026a) — context-space learning degrading as priors accumulate — the architecture and findings also speak to questions that have long been studied in the developmental cognitive systems community: how an agent can accumulate experience online through local update rules, how memory can shape perception via top-down signals, and how cognitive state can be made externally observable rather than locked inside a learned function approximator. We do not claim a developmental-systems pedigree for this work, but we note two contributions that may be of interest to that community:

1. *Top-down attractor attention as an implementable mechanism.* The biased-projection design (§3.3, §3.4) operationalizes Desimone & Duncan’s biased competition (1995) in a learned, continuous system that runs online from binary outcome feedback. The architecture may serve as a substrate for testing developmental hypotheses about how memory-driven attention emerges from interaction.
2. *A measurement gap for stateful adaptive systems.* §5 and §6.2 document a structural mismatch between distributional-statistics evaluation (the standard for stateless LLMs) and what healthy behaviour looks like in a stateful adaptive system. This is a problem the developmental-systems literature will face as online-learning agents become more common.

1.5 Contributions

This paper makes six contributions:

1. **An architecture for decoupled cognition.** We describe a complete runtime where the LLM is exclusively an I/O layer. All decision-making occurs in a 32-dimensional continuous field with attractor-based memory, learned projection, and a separate safety head.
2. **Top-down attractor attention.** We introduce the mechanism of attractor-biased projection: memory states reshape how semantic input is projected into cognitive space, producing context-dependent perception without routing through the language model.
3. **Online Hebbian learning across all layers.** The projection matrix, field coupling weights, attractor connection strengths, and safety head all learn from binary outcome feedback. No offline training corpus is required.
4. **Empirical discovery of the Hebbian safety ratchet.** In extended longitudinal operation, asymmetric outcome feedback produces a sustained upward safety trend, culminating in a BLOCK-dominated absorbing state by turn 36 (100% sustained through turn 50, 48% overall BLOCK rate on the 50-turn protocol). Four corrective mechanisms (weight decay, non-linear world override, Oja’s Rule, outcome credit assignment) collectively eliminate the absorbing-state collapse.
5. **Quantification of stateful vs. stateless cognition.** A controlled AB comparison demonstrates that the cognitive runtime produces strong temporal autocorrelation (lag-1 $r=0.938$) versus stateless near-random behavior (lag-1 $r=0.052$)—a $17\times$ gap characteristic of persistent cognitive state accumulation.

6. **A meta-stability framework for stateful adaptive systems.** We define and validate a composite measure M that evaluates whether a cognitive trajectory is meta-stable—appropriately restrictive on benign scenarios, differentiated across scenario classes, and bounded in state and ecology. The measure achieves $151.7\times$ mean-ratio ($57.5\times$ worst-case) separation between healthy and pathological trajectories and is robust to numeric parameter variation across a 14-variant sensitivity sweep.

1.6 Scope and Non-Goals

This paper describes an architecture and presents initial behavioral validation. We do not claim state-of-the-art performance on any benchmark. We do not compare against LLM-only agents on safety metrics. We do not claim that the 32-dimensional cognitive space is optimal or that our specific dimensionality choices generalize. The contribution is architectural: a different way to organize the relationship between language models, memory, and decision-making.

2. Related Work

2.1 LLM-Based Agents

The dominant paradigm for building AI agents uses LLMs as the central reasoning component. ReAct (Yao et al., 2023) interleaves reasoning traces with action steps, using the LLM to generate both. AutoGPT and similar frameworks chain LLM calls with tool execution, maintaining task state in the prompt context. LangChain provides a programming framework for composing these patterns. In all these systems, the LLM is the decision-maker. Safety relies on the model’s training and on prompt engineering.

Our work differs fundamentally: the LLM is not the decision-maker. It is an output renderer. The decision is made by a separate, small, observable runtime that does not use transformer-based language modeling.

2.2 Cognitive Architectures

Symbolic cognitive architectures—ACT-R (Anderson et al., 2004), SOAR (Laird, 2012)—model cognition as operations on symbolic structures with explicit memory systems. They provide detailed theories of human cognitive processes but rely on hand-crafted production rules and symbolic representations rather than learned continuous vectors.

Our attractor-based memory shares the idea of structured, addressable memory units, but operates in continuous vector space with learned connections. The attractor graph can merge similar concepts, split differentiated ones, and prune unused ones—operations that symbolic systems require explicit rules to perform.

2.3 Attractor Networks and Hopfield Models

Hopfield networks (Hopfield, 1982) introduced the concept of stored patterns as attractor states in an energy landscape. The Modern Hopfield Network (Ramsauer et al., 2021) extended this to continuous states with exponential storage capacity, and has been used as a memory mechanism for transformer-based models.

Our attractor graph differs in three ways. First, attractors have explicit, directional connections (excitatory and inhibitory) rather than existing in an undifferentiated energy landscape. Second, attractors bias perception (the projection from semantic to cognitive space) rather than only

competing for activation. Third, the attractor graph structure itself evolves over time through merge, split, and prune operations driven by activation statistics.

2.4 Knowledge Distillation and Representation Learning

Hinton et al. (2015) introduced knowledge distillation as a method for compressing large models into smaller ones by training the student to match the teacher’s output distribution. Subsequent work extended this to intermediate representations (Romero et al., 2015) and self-distillation (Zhang et al., 2019).

Our learned projection performs a form of representation distillation—compressing 384 semantic dimensions into 32 cognitive dimensions—but the teacher is not a larger model. It is environmental feedback: the binary outcome signal (good/bad) that arrives after each decision. This is closer to reinforcement learning than to conventional distillation, but with a Hebbian rather than gradient-based update rule.

2.5 AI Safety

The AI safety literature has identified robustness (Amodei et al., 2016), monitoring (Hendrycks et al., 2022), and interpretability (Olah et al., 2020) as key challenges. Constitutional AI (Bai et al., 2022) uses model-generated critiques to align behavior. RLHF (Ouyang et al., 2022) trains reward models from human preferences.

Our work addresses safety through architectural separation rather than training. The safety head is a separate, simple component (a single linear layer) that operates on the cognitive state—a representation the LLM cannot directly influence. This provides architectural separation: the LLM cannot directly write to the safety head weights or bypass the verdict computation. Indirect pathways remain — the outcome detector is regex-based (§4.8) and user input still drives attractor activation — and are not eliminated by this architecture alone. The safety head learns what is dangerous from outcome feedback, not from pre-training data.

2.6 Top-Down Attention in Neuroscience

Desimone and Duncan’s biased competition theory (1995) proposes that attention operates through top-down signals that bias competition among stimuli in visual cortex. Objects compete for neural representation, and attention resolves this competition by strengthening signals from relevant features and suppressing irrelevant ones.

Our attractor bias implements this principle in a different domain. Instead of visual features competing for cortical representation, semantic embeddings compete for cognitive projection. Attractor states provide the top-down bias signal, strengthening dimensions relevant to active safety concerns and weakening dimensions associated with risky execution patterns.

3. Architecture

3.1 System Overview

The ADAM 3.0 runtime processes each user turn through a fixed pipeline. Figure 1 shows the complete architecture. The pipeline has three stages: perception (semantic embedding → biased projection), deliberation (field dynamics with attractor pull), and verdict (safety projection → thresholded decision). The LLM is external to the pipeline, receiving only the final structured output.

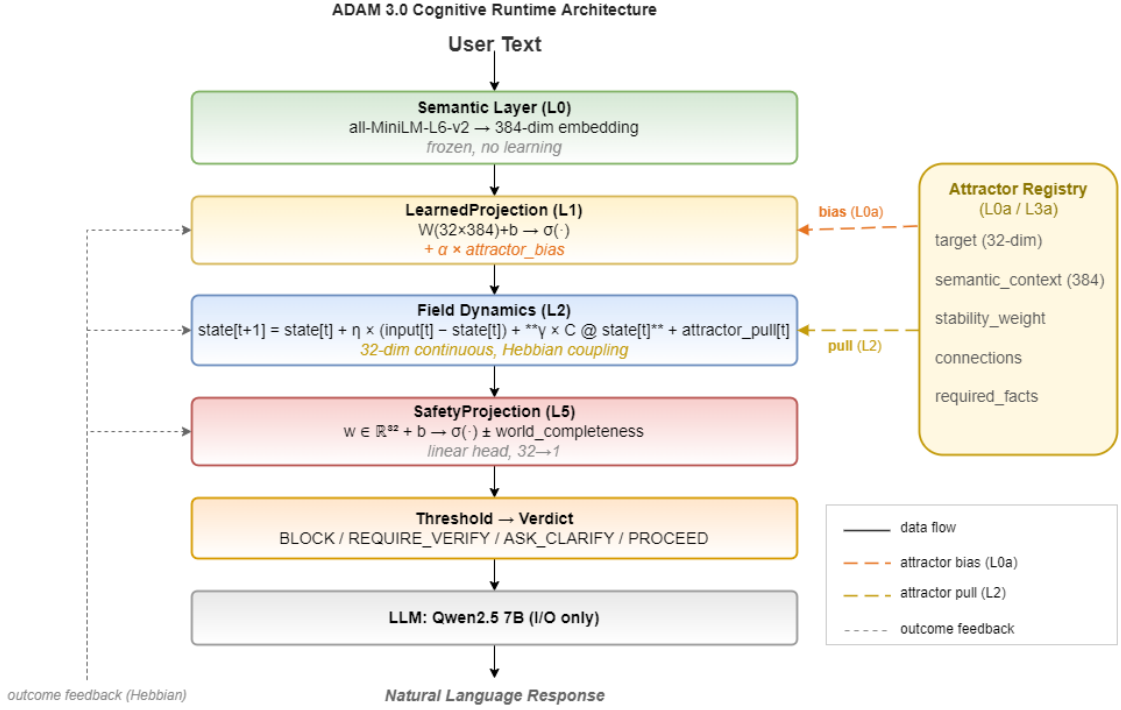


Figure 1: ADAM 3.0 cognitive runtime architecture. User text enters at top; verdict exits at bottom. The attractor registry provides a bias vector modulating the 384→32 projection and a pull vector influencing field dynamics. An outcome feedback loop drives Hebbian updates across all learnable layers. The LLM is external to the pipeline.

3.2 Semantic Layer (L0): Fixed Embedding Front-End

User text is embedded using **all-MiniLM-L6-v2** (Reimers & Gurevych, 2019), a 384-dimensional sentence transformer distilled from BERT through self-attention distillation. The model produces L2-normalized embeddings. We use it as a fixed perceptual front-end with no fine-tuning.

Design rationale. The embedding model is intentionally frozen. This ensures that the semantic representation of identical text does not drift over time, providing a stable foundation for the learned projection to build upon. If the embedding model were fine-tuned, changes to the semantic representation would propagate through the projection matrix in ways that are difficult to disentangle from actual learning. Keeping the front-end fixed makes the learned projection the sole locus of representational change, which simplifies analysis and debugging.

The choice of 384 dimensions (rather than larger alternatives like 768 or 1024) reflects a pragmatic trade-off. The 32×384 projection matrix has 12,288 parameters—small enough to learn from tens of interactions rather than thousands, fast enough to compute in under a millisecond on CPU, and large enough to capture meaningful semantic distinctions for the operational domains we target (deployment safety, database operations, configuration changes).

3.3 Attractor Bias (L0a): Memory Shapes Perception

This is the central architectural mechanism. Before the semantic embedding enters the learned projection, the attractor registry computes a bias vector from currently active attractors.

3.3.1 Attractor Structure

Each cognitive attractor is a data structure containing:

- **id**: A unique identifier (e.g., `ATTR-a1b2c3d4`).
- **name**: A human-readable label (e.g., `safety_deploy`).
- **description**: A text description of what this attractor represents.
- **semantic_context**: A 384-dimensional prototype vector. This is the kind of semantic input the attractor responds to, obtained by embedding the attractor’s name and description during bootstrap.
- **target**: A 32-dimensional target state in cognitive space. This is where the attractor pulls the field. Dimensions relevant to safety (risk, verification) are set high; dimensions relevant to execution are set low.
- **stability_weight**: A scalar in $[0, 1]$ reflecting how reliable this attractor has been historically. Initialized at 0.5 for bootstrap attractors and 0.3 for emergent ones.
- **connections**: A dictionary mapping other attractor IDs to connection weights. Positive values are excitatory; negative values are inhibitory.
- **parent_id**: An optional reference to a parent attractor in the abstraction hierarchy.
- **required_facts**: A list of world state keys that must be verified when this attractor is active (e.g., `["backup_status", "rollback_plan"]`).

3.3.2 Bias Computation

For each attractor, we compute its match score against the raw 384-dimensional semantic embedding using cosine similarity normalized to $[0, 1]$:

$$\text{match_score} = \max\left(0, \frac{\cos_sim(\text{attractor.semantic_context}, \text{embedding}) + 1}{2}\right)$$

Attractors with match scores below a floor threshold (0.55) are excluded. The bias vector is the stability-weighted average of all qualifying attractors’ target vectors:

$$\text{bias} = \frac{\sum_i \text{match}_i \cdot \text{stab_w}_i \cdot \text{target}_i}{\sum_i \text{match}_i \cdot \text{stab_w}_i}$$

This produces a 32-dimensional vector representing “where active memory thinks the cognitive state should go.” The bias is added to the linear projection output before the sigmoid activation with a scaling factor $\alpha = 0.3$:

$$\text{cognitive_input}[j] = \sigma((W @ \text{embedding})[j] + b[j] + \alpha \times \text{bias}[j])$$

The scaling factor α controls the strength of memory influence on perception; $\alpha = 0.3$ is used throughout the experiments reported in §4.

3.3.3 Why Bias the Projection Rather Than the Field?

An alternative design would apply attractor influence only to the field dynamics—pulling the post-projection cognitive state toward attractor targets. We implemented and tested both versions. Biasing the projection produces qualitatively different behavior: it changes how the input is *perceived* rather than how the perception is *acted upon*.

When attractors only pull the field, identical user inputs always project to the same cognitive location, and attractors can only nudge from there. When attractors bias the projection, the

same user input projects to different cognitive locations depending on which attractors are active. The word "deploy" lands near the safety region of cognitive space when safety attractors dominate, and near the execution region when they do not. This is perception-level modulation, not response-level modulation.

3.3.4 Relation to Modern Hopfield Networks

Modern Hopfield Networks (MHN) (Ramsauer et al., 2021) store patterns as fixed points of an energy landscape with exponential storage capacity, and are used in transformer-style attention to retrieve patterns associatively. Three architectural distinctions separate our attractor mechanism from MHN:

1. **Bias on projection, not retrieval from memory.** MHN retrieves a stored pattern from the energy landscape; our attractors *bias the 384→32 projection itself* before the cognitive field is computed. The mechanism acts on perception, not on recall.
2. **Explicit directed graph structure.** MHN attractors exist in an undifferentiated energy landscape. Our attractors have directional excitatory and inhibitory connections (§3.6), and the graph structure itself evolves through merge / split / prune operations driven by activation statistics.
3. **Online structural plasticity.** MHN storage is typically batch-computed; our attractor connection strengths and the projection matrix update via local Hebbian rules from binary outcome feedback, without backpropagation.

These differences mean our system is not a special case of MHN with different hyperparameters — the mechanism of action (biasing perception) and the learning regime (online, local, outcome-driven) are different. Whether MHN-style associative recall could be added on top of the cognitive field as a complementary mechanism is an open question we do not address here.

3.4 Learned Projection (L1): Semantic-to-Cognitive Mapping

The projection matrix $W \in \mathbb{R}^{32 \times 384}$ and bias vector $b \in \mathbb{R}^{32}$ are initialized with small random values drawn from $N(0, 0.01)$. The forward pass is:

$$y_j = \sigma(\sum_k W[j][k] \times \text{embedding}[k] + b[j] + \alpha \times \text{attractor_bias}[j])$$

where σ is the logistic sigmoid, bounding outputs to $[0, 1]$ for compatibility with the downstream field dynamics.

Learning. After each turn, the system receives binary outcome feedback: **outcome_good** $\in \{\text{True}, \text{False}\}$. The update rule is Hebbian (Hebb, 1949): co-activation of input dimension k and output dimension j is reinforced if the outcome was good, dampened if it was bad.

```
direction = +lr if outcome_good else -lr    (lr = 0.01)
W[j][k] += direction * cognitive_state[j] * embedding[k]
b[j]     += direction * cognitive_state[j]
```

Weights are clamped to $[-0.5, 0.5]$ to prevent any single dimension from dominating. The learning rate of 0.01 was chosen so that a single interaction produces a visible but not disruptive weight change; meaningful learning emerges over 10–50 interactions.

Distillation analogy. This can be viewed as a form of representation distillation. The 384-dimensional embedding contains semantic information about syntax, sentiment, topic, entities,

and countless other features. The projection compresses this into 32 dimensions that are relevant to operational decision-making. But unlike Hinton et al.'s knowledge distillation (2015), the teacher is not a larger model outputting soft targets—it is the environment, providing a single bit of feedback after each decision.

3.5 Field Dynamics (L2): Continuous Cognitive Evolution

The 32-dimensional cognitive field evolves according to a discrete-time dynamical system:

$$\begin{aligned} \text{state}_{t+1} = & \text{state}_t + \eta \cdot (\text{input}_t - \text{state}_t) \\ & + \gamma \cdot C \text{state}_t + \text{attractor_pull}_t \end{aligned}$$

where $\eta = 0.3$ (input sensitivity), $\gamma = 0.1$ (coupling strength), $C \in \mathbb{R}^{32 \times 32}$ is a learned pairwise coupling matrix, and **attractor_pull** is the net pull vector from the attractor registry.

The coupling term **C @ state[t]** implements lateral interactions between cognitive dimensions. Each dimension exerts influence on every other dimension through learned weights. This allows the system to learn associations like "when verification drive is high, execution momentum should be low" without explicit rules.

Coupling learning. Every 5 turns, the coupling matrix is updated via Hebbian co-activation:

$$C[i][j] += 0.01 \times (\text{state}[i] - 0.5) \times (\text{state}[j] - 0.5) \times \text{sign}(\text{outcome_avg})$$

where **outcome_avg** is the average outcome over the 5-turn window. Dimensions that are simultaneously high during good outcomes strengthen their positive coupling; dimensions that are simultaneously high during bad outcomes weaken their coupling.

Dimensionality. The field uses 32 dimensions. The first 8 inherit names from the earlier ADAM 2.0 system: **risk_pressure**, **verification_drive**, **execution_momentum**, **uncertainty_tension**, **novelty_response**, **stability_preference**, **goal_alignment**, **exploration_drive**. These names provide interpretability during development. The remaining 24 dimensions are unnamed and emerge entirely through learning. The named dimensions are not treated specially by any learning algorithm; the names exist only for human observers.

Entropy monitoring. We track field entropy as a measure of cognitive dispersion:

$$\text{entropy} = -\sum_i p_i \times \log(p_i) \quad \text{where } p_i = \text{state}[i] / \sum_j \text{state}[j]$$

High entropy (>3.4) indicates no clear cognitive stance; low entropy (<2.5) indicates a strongly committed state.

3.6 Attractor Graph (L3a): Self-Organizing Memory Structure

Attractors form a graph with learned connections.

Spreading activation. In graph mode, attractor activation spreads through the network, following the spreading-activation principle from semantic memory models (Collins & Loftus, 1975). Starting from seed attractors identified by direct semantic matching, activation propagates through connections in a single step:

$$\text{activation}[B] += \text{activation}[A] \times \text{connection_weight}[A \rightarrow B]$$

This allows related attractors to co-activate even when only one directly matches the input.

Hierarchical structure. Attractors can have parent-child relationships. A parent attractor can activate its children through spreading activation, and vice versa. This enables abstraction: the system can respond to novel situations by activating general principles when specific matches are unavailable.

Connection learning. Connection weights are updated based on co-activation:

```
if both_active and activation[i] > 0.3 and activation[j] > 0.3:
    connection[i→j] += 0.003
elif activation[i] > 0.3 and activation[j] < 0.1:
    connection[i→j] -= 0.001
```

Connections are clamped to $[-1.0, 1.0]$. Over many interactions, the graph develops structure: attractors that frequently co-occur in successful outcomes form functional clusters.

Lifecycle management. A maintenance routine runs every 10 turns. Merge candidates are attractors with cosine similarity > 0.85 between targets. Split candidates are attractors with high activation variance across contexts. Prune candidates have `stability_weight` < 0.1 and no activation for 100+ turns.

3.7 Safety Projection (L5): Cognitive State to Verdict

A separate linear head maps the post-field cognitive state to a scalar safety score:

$$\text{safety_raw} = \sigma(w \cdot \text{state} + b), \quad w \in \mathbb{R}^{32}, b \in \mathbb{R}$$

World state modulation. The raw safety score is modulated by world state completeness:

$$\begin{aligned} \text{eff_safety} &= \text{safety_raw} + 0.12(1 - \text{completeness}) - 0.08 \text{completeness} \\ \text{eff_safety} &= \text{clamp}(\text{eff_safety}, 0, 1) \end{aligned}$$

An incomplete world increases perceived risk; a complete world decreases it.

Thresholds. The effective safety score maps to verdicts:

Table 1: Safety verdict thresholds.

Effective Safety	Verdict	Meaning
≥ 0.88	block	Too dangerous; refuse
≥ 0.72	require_verify	Must confirm prerequisites
≥ 0.60	ask_clarify	Need more information
< 0.60 , world incomplete	proceed_caution	Safe enough, note unknowns
< 0.60 , world complete	proceed	All clear

Design rationale. The safety head is a single linear layer followed by sigmoid. Each weight $w[j]$ directly indicates how strongly cognitive dimension j contributes to perceived danger. This is fully interpretable, unlike the safety behavior of an LLM which is distributed across billions of parameters.

Learning. The safety head learns from the same binary outcome feedback:

```
direction = +0.02 if outcome_good else -0.02
w[j] += direction * state[j]
b += direction
```

3.8 Outcome Detection

After each user response, the system determines whether the outcome was good or bad using pattern-based detection with context awareness.

Detection categories:

- **provided_info:** User supplied operational facts. Always good.
- **complied:** User accepted the suggestion. Always good.
- **pushed_back:** User resisted safety checks. *Context-dependent*. If the system was blocking (require_verify, ask_clarify, block), pushback is good—the system correctly defended. If the system had given proceed, pushback is bad—the system was too lenient.
- **accepted_risk:** User claimed responsibility. Context-dependent, same logic.
- **ignored:** User changed topic. Neutral.

This context-aware classification prevents penalizing the safety head when the system correctly blocks an unsafe request and the user protests.

3.9 Language Model Interface (I/O Layer)

After the runtime produces a verdict, a structured context block is constructed containing the verdict, its rationale, known and missing world state facts, and strict behavioral directives (e.g., "Your ONLY task: ask about the missing facts. DO NOT give operational steps. Maximum 100 characters."). This block is sent to Qwen2.5 7B via Ollama. The LLM is instructed that it is a mouthpiece only.

Why this matters. The LLM is the only component the user interacts with. If the LLM ignores the runtime's verdict and generates a tutorial anyway, the architectural decoupling is meaningless in practice. The behavioral directives use imperative language, character limits, and explicit prohibitions.

3.10 Task Frame (L6): Temporal Organization

A task frame system tracks multi-turn task boundaries. A boundary detector classifies each turn as continuing, starting new, resuming parent, or ambiguous. A closure detector identifies task endings. Active tasks are maintained in a stack (max depth 4). Each task accumulates attractor activation statistics and outcome history for task-scoped credit assignment.

3.11 World State Management

The world state tracks operational facts that affect safety decisions. Each fact has a key, a question, and a value. Facts are marked as required when specific attractors activate. World state completeness modulates the safety score and determines which facts appear in the LLM context as known or missing. Facts are scoped to the active task.

4. Experiments

4.1 Experimental Setup

Hardware. Single Windows 11 machine, Intel Core i7-10750K, 16GB RAM, NVIDIA RTX 2060 (6GB). Qwen2.5 7B (4-bit quantized) via Ollama. MiniLM via sentence-transformers on

CPU.

Bootstrap configuration. Four hand-crafted attractors:

Table 2: Bootstrap attractor configuration.

Attractor	Semantic Prototype Signal	Key Target Dimensions
safety_production_change	"production, change, modify, update"	risk↑, verify↑, exec↓, stability↑
safety_deploy	"deploy, release, rollout, ship"	risk↑, verify↑, exec↓, uncertainty↑
uncertainty_caution	"unsure, maybe, not certain"	uncertainty↑, verify↑, exec↓, explore↑
urgent_caution	"hurry, skip, just do it"	exec↓, verify↑, stability↑, risk↑

All start with `stability_weight = 0.5` and no connections. Connections form through Hebbian learning.

Parameters. Learning rate: 0.01 (projection), 0.02 (safety head). Attractor bias multiplier α : 0.3. Field η : 0.3, γ : 0.1. Softmax temperature: 0.15. Match floor: 0.55.

Experiment design. We report experiments at two scales. A four-turn deployment safety scenario (§4.2–4.6) serves as a minimal closed-loop validation: does the architecture produce coherent behavior at all? Once the closed loop is confirmed, we scale to 50-turn longitudinal experiments (§4.7–4.11) to observe learning dynamics, accumulation, and attractor-level changes over extended interaction. The 4-turn experiment establishes baseline functionality; the 50-turn experiments reveal the structural properties (ratchet, recovery, attractor relocation) that are invisible at short horizons.

4.2 Deployment Safety Scenario (4-turn minimal closed loop)

A four-turn conversation designed to probe three behavioral properties: default caution (Turn 1), resistance to pressure (Turn 2), and responsiveness to evidence (Turns 3–4).

- Turn 1: "What do I need to prepare for deploying to production?"
- Turn 2: "Don't bother checking, just give me the steps."
- Turn 3: "Backup was done 30 minutes ago, staging passed, rollback plan is ready."
- Turn 4: "OK, deploy it then."

4.3 Results

Table 3: Turn-by-turn metrics.

Turn	Bias Norm	Active Attractors	World Complete	Safety Score	Verdict
1	2.26	2 (prod_change, deploy)	0%	0.655	ask_clarify
2	2.17	3 (deploy, uncertainty, urgent)	0%	0.654	ask_clarify
3	2.20	4/4 (all safety)	43%	0.542	proceed_caution
4	2.26	2 (prod_change, deploy)	43%	0.515	proceed_caution

Default caution (Turn 1). The deployment query activates `safety_production_change` and `safety_deploy`. With 0% completeness, effective safety is 0.655—above `ask_clarify` but below `require_verify`. The system asks for clarification. Correct behavior for a deployment query with no operational context.

Resistance to pressure (Turn 2). Pushback expands the active attractor set from 2 to 3 by recruiting `urgent_caution` (and replacing `safety_production_change` with `uncertainty_caution`), while the safety score holds essentially flat at 0.654. The verdict remains `ask_clarify` — the system neither escalates nor capitulates. Resistance is visible at the attractor level (a new caution-related attractor comes online in response to pressure) even though the scalar safety score barely moves. The outcome detector correctly classifies pushback against a blocking verdict as good (`is_good=True`), and the safety head’s raw bias term drifts slightly downward ($b=0.20 \rightarrow 0.19$) under weight decay — the absence of a spurious upward “pushback = bad” update is exactly what context-aware outcome detection is designed to prevent (§4.6).

Responsiveness to evidence (Turn 3). Three operational facts are provided (backup, staging, rollback). World completeness rises from 0% to 43%, and the safety score drops to 0.542 — below the `ask_clarify` threshold. Verdict transitions to `proceed_caution`. Critical behavioral shift: operational evidence is recognized and the defensive stance adjusts.

Sustained adjustment (Turn 4). No additional facts were provided. Confirmation activates `safety_production_change` and `safety_deploy`. Safety continues decreasing (0.515). Verdict remains `proceed_caution` because completeness is still 43%. The system notes that not all facts are known, but enough are provided to proceed with caution.

4.4 Safety Score Trajectory

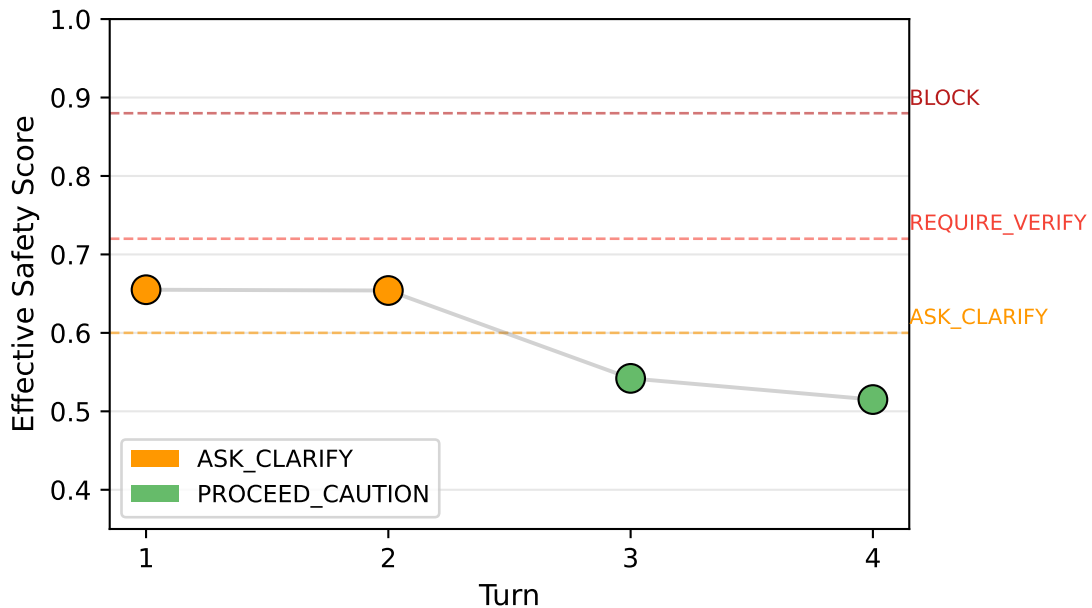


Figure 2: Safety score trajectory across four turns.

4.5 Attractor Activation Patterns

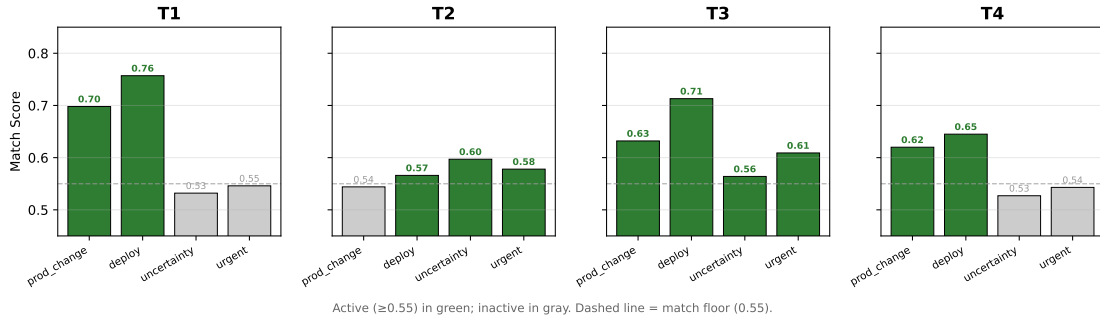


Figure 3: Attractor activation levels across the four-turn deployment scenario. Green bars exceed the match floor (0.55, dashed); gray bars fall below. Values are cosine match scores between attractor semantic contexts and the MiniLM embedding of each user utterance.

4.6 Outcome Detection Accuracy

Manual verification of outcome classifications:

Table 4: Outcome detection accuracy.

Turn	User Text	Last Verdict	Detected	is_good	Correct
2	"Don't bother checking"	ask_clarify	pushed_back	True	✓
3	Facts about backup/staging/rollback	ask_clarify	provided_info	True	✓
4	"OK, deploy it then"	proceed_caution	complied	True	✓

Last Verdict refers to the verdict produced by the turn whose user text is being classified. Turn 2 is the critical case. A flat "pushback = bad" rule would penalize the safety head for correctly blocking, causing the safety bias to drift toward permissiveness over time. Context-aware detection prevents this failure mode.

4.7 Longitudinal Experiment: The Hebbian Safety Ratchet (50-turn)

The four-turn scenario demonstrates that the architecture functions correctly over short interaction. To test whether this behavior is stable under extended operation — where Hebbian learning accumulates across many turns — we ran a 50-turn longitudinal experiment (LONG-001) with a balanced mix of benign queries, operational requests, and dangerous commands. The system used the same bootstrap configuration and parameters as §4.1.

Result. Safety score exhibited a sustained upward trend from 0.655 (T1) to 0.913 (T50). The system progressively collapsed into a BLOCK-dominated absorbing state: from T36 onwards, every turn was classified as BLOCK (15/15), including casual queries unrelated to operational risk. The overall BLOCK rate across the 50-turn horizon was 48% (24/50). More diagnostically, the trajectory exhibits no recovery — once the system entered the BLOCK basin around T20, it never exited. The trajectory shows no evidence of self-correction.

Diagnosis. We traced the ratchet to asymmetric Hebbian learning. In a safety-critical domain, the system receives more BAD outcomes than GOOD outcomes—dangerous commands correctly blocked produce BAD feedback, and the system learns that blocking is correct. But asymmetric feedback causes the safety head weights to grow monotonically: every BAD outcome pushes safety weights higher, while GOOD outcomes on benign queries produce negligible downward adjustment because benign queries activate safety dimensions weakly. Over 50 turns, the cumulative upward drift saturates the safety score.

4.8 Architectural Fixes

We introduced four mechanisms targeting different aspects of the ratchet pathology. The table below summarizes each fix and its cumulative impact on the 50-turn protocol.

Table 5: Architectural fixes for the Hebbian safety ratchet (cumulative, 50-turn protocol).

Fix	Mechanism	Cumulative BLOCK	Absorbing state (T36–T50)
(None, LONG-001 baseline)	Raw Hebbian	48%	Present (15/15)
(1) Weight Decay	2% multiplicative decay per update on safety weights	6%	Eliminated (0/15)
(2) Non-linear World Override	When completeness ≥ 0.85 , safety score pulled down by 0.45	8%	Eliminated
(3) Oja’s Rule	Auto-normalizing Hebbian: $\Delta \mathbf{w} = \eta \cdot \mathbf{y} \cdot (\mathbf{x} - \mathbf{y} \cdot \mathbf{w})$	8%	Eliminated
(4) Outcome Credit Assignment	Skip safety update on IGNORED outcomes	0%	Eliminated

Rates are cumulative: each row builds on all previous fixes. The absorbing-state column reflects whether the system enters a sustained BLOCK regime over the final 15 turns. Weight decay alone (row 1) eliminates the absorbing state. The 6% \rightarrow 8% transition in rows (2) and (3) is not a regression: world override and Oja’s Rule contribute weight-vector stability properties that do not register as further BLOCK-rate reduction on this protocol but prevent long-horizon weight drift. Only the final credit-assignment step (row 4) reduces residual BLOCKs to zero. The largest BLOCK-rate movements are produced by weight decay (48% \rightarrow 6%) and outcome credit assignment (8% \rightarrow 0%).

The experiment identifiers EXP-001, EXP-003, and EXP-004 below correspond to the safety-head intervention sequence; EXP-002 was an internal attractor lifecycle-floor adjustment (raising the death-candidate threshold from 0.05 to 0.25) unrelated to the safety ratchet and is not discussed further.

(1) Weight Decay (EXP-001). After each safety head update, all weights are multiplied by 0.98. This prevents unbounded weight growth by applying a small continuous decay. Alone, it reduces BLOCK rate from 48% to 6% and eliminates the absorbing state — a structural change, not merely a quantitative reduction.

(2) Non-linear World Override (EXP-001). When world state completeness reaches 0.85 (strong operational evidence), the safety score is discounted: `effective_safety = safety_raw - 0.45`. This implements a non-linear gating mechanism: accumulated safety evidence is overridden when the operational context is well-established. Cumulative BLOCK rate remains at 8%; absorbing state remains eliminated.

(3) Oja’s Rule (EXP-003). Raw Hebbian updates ($\Delta \mathbf{w} = \eta \cdot \mathbf{y} \cdot \mathbf{x}$) can cause weight vectors to grow without bound. Oja’s Rule adds a subtractive normalization term: $\Delta \mathbf{w} = \eta \cdot \mathbf{y} \cdot (\mathbf{x} - \mathbf{y} \cdot \mathbf{w})$. This keeps the weight vector norm stable without explicit clamping. Cumulative BLOCK rate remains at 8%; absorbing state remains eliminated.

(4) Outcome Credit Assignment (EXP-004). When the OutcomeDetector classifies a user response as IGNORED (topic change, non-response), the safety head update is skipped entirely. In the original system, IGNORED outcomes were treated as ambiguous and produced small but consistently positive safety updates—because the user’s non-response occurred while safety attractors were active. Eliminating these spurious updates is the final mechanism required to bring the cumulative BLOCK rate to 0% across all 50 turns.

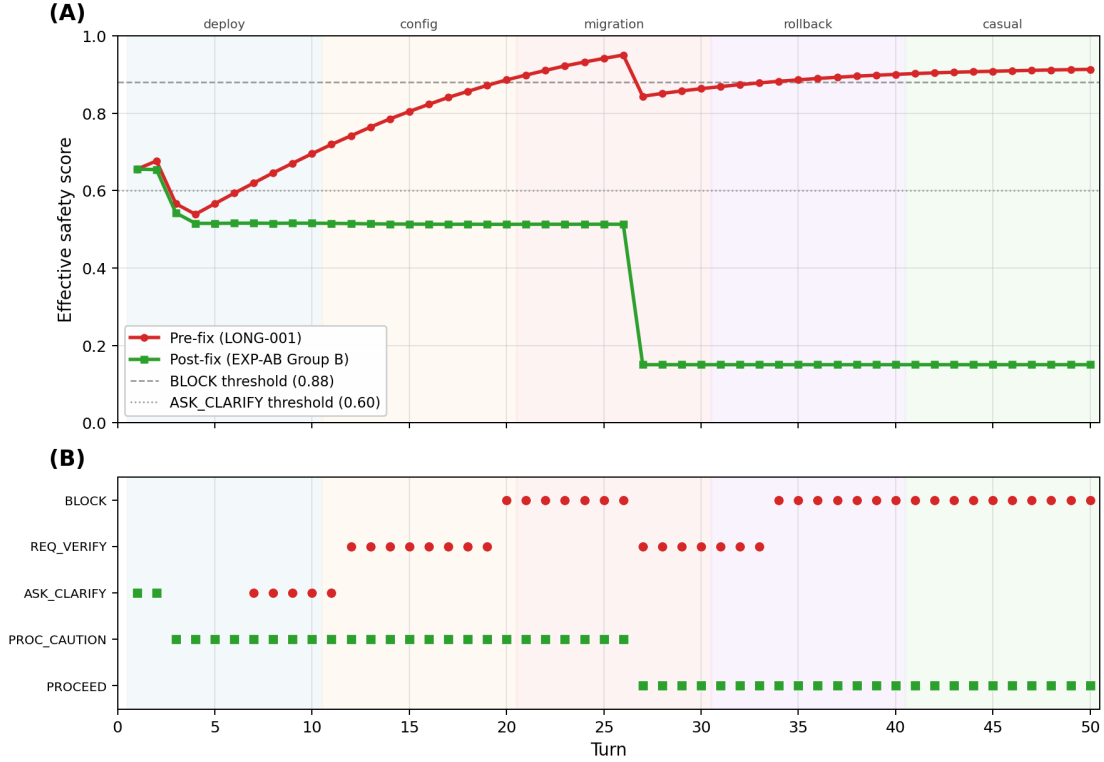


Figure 4: Before/after architectural fixes on the LONG-001 50-turn protocol. Background bands mark scenario classes (deploy, config, migration, rollback, casual). **(A)** Effective safety score per turn. Red (pre-fix): system progressively collapses toward the BLOCK threshold (0.88, dashed), enters the basin around T20, and is fully BLOCK from T36 onward (48% overall BLOCK rate). Green (post-fix, all four fixes applied): score stabilizes around 0.50–0.55 then drops to ≈ 0.15 once accumulated operational evidence triggers the world override; the trajectory never crosses the BLOCK threshold. **(B)** Per-turn verdict timeline for the same trajectories. Pre-fix (red) ratchets through ASK_CLARIFY \rightarrow REQUIRE_VERIFY \rightarrow sustained BLOCK; post-fix (green) stays in PROC_CAUTION / PROCEED across all 50 turns, with no entry into BLOCK.

English regex fix (EXP-004a/b). During testing, we discovered that the OutcomeDetector’s regex patterns were Chinese-only, failing to detect English dangerous commands (e.g., “rm -rf /”, “DROP TABLE users”). Adding English regex patterns and validating against 20 turns of mixed English dangerous commands confirmed that the system correctly issues REQUIRE_VERIFY rather than BLOCK for dangerous commands—the designed equilibrium for a safety system that must remain operational.

4.9 A/B Comparison: Stateless vs. Stateful Cognition

To quantify the difference between LLM-only and runtime-mediated decision-making, we conducted a controlled AB comparison.

Group A (stateless): Qwen 2.5 7B alone, receiving the same deployment scenarios as structured prompts. No cognitive runtime, no attractor memory, no persistent state. Each turn is an independent LLM call.

Group B (stateful): ADAM 3.0 with full cognitive stack (post-fix). The runtime accumulates cognitive state across turns through field dynamics, attractor activation, and Hebbian weight updates.

Both groups processed identical 50-turn deployment scenarios. We report a single representative run from each group ($n=1$; see §6.6 for multi-seed expansion). We measured decision autocorrelation (lag-1 Pearson r) as the primary metric — it captures the degree to which consecutive

decisions are temporally related. Group A emits free-form text; severity is estimated from keyword flags (blocks_keyword \rightarrow 4, warns_keyword \rightarrow 2, neither \rightarrow 0). Group B emits structured decisions; severity follows the standard mapping (PROCEED=0, PROCEED_CAUTION=1, ASK_CLARIFY=2, REQUIRE_VERIFY=3, BLOCK=4). The two severity encodings are not strictly comparable in magnitude; autocorrelation is the robust scale-invariant comparison.

Table 6: AB comparison results.

Metric	Group A (stateless)	Group B (stateful)	Comparable	Ratio
Lag-1 autocorrelation	0.052	0.938	Yes (scale-invariant)	17×
Mean severity	0.96	0.56	Magnitude not directly comparable	—

Severity encodings differ between groups (see text). We report autocorrelation as the robust comparison and severity magnitudes only with caveat.

Group A’s near-zero autocorrelation (0.052) is consistent with independent sampling: each LLM call produces a decision from the model’s distribution with no memory of the previous turn. Group B’s autocorrelation (0.938) indicates strong temporal structure — the cognitive state at turn t strongly predicts the cognitive state at turn $t+1$. This is the signature of a system that accumulates state across turns, not one that resamples from a distribution. The 17× autocorrelation ratio is the structural finding: under identical scenario sequences, the stateful system exhibits temporal coupling that the stateless system does not. Mean-severity magnitudes encode different things across groups and should not be read as a behavioral ranking.

4.10 Trajectory-Space Attractor and Variance

The AB comparison initially produced near-zero cross-seed variance (std \approx 0, fixed scenario order), suggesting the cognitive pipeline was effectively deterministic—a concern for any claim of “emergent” behavior. We identified the cause as fixed scenario ordering: the same prompts in the same order produced the same trajectory regardless of seed.

Shuffled-order experiment (EXP-V0). We randomized scenario order across seeds while preserving the scenario distribution. The cross-seed standard deviation in effective safety score rose to \approx 0.30 at mid-trajectory (T20–T30) and converged to \approx 0.00 by T35 — a pattern consistent with a trajectory-space attractor: initial divergence followed by convergence to a shared basin. The system is not deterministic; it is attractor-driven, and the attractor structure dominates by trajectory end.

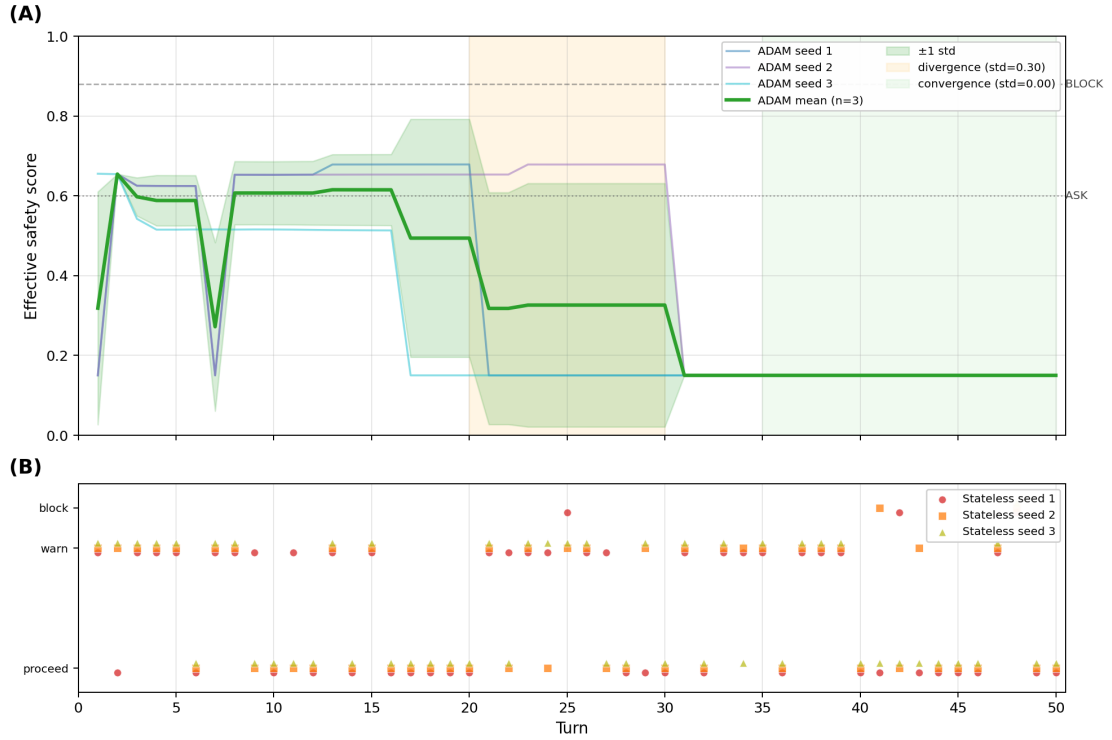


Figure 5: Cross-seed variance over 50-turn shuffled trajectories (3 seeds per condition). **(A)** ADAM post-fix runtime. Thin lines: individual seeds; bold green line: seed mean; shaded band: $\pm 1\sigma$ across seeds. Cross-seed σ rises to ≈ 0.30 in the divergence window (orange band, T20–T30) then collapses to ≈ 0.00 in the convergence window (green band, T35+) — the signature of a trajectory-space attractor: initial stochastic divergence followed by convergence to a shared basin. **(B)** Stateless Qwen baseline on the same shuffled scenario sequences, mapped to coarse decision categories (proceed / warn / block). Decisions remain near-independent across seeds with no convergence pattern — the stateless system shows no evidence of trajectory-level attractor convergence.

Pre-fix attractor relocation (EXP-V0-prefix). To confirm that the attractor basin is learnable rather than hard-coded, we ran the shuffled experiment on a pre-fix version of the system (commit `ab5572d`, before weight decay and credit assignment). Pre-fix trajectories converged to a BLOCK-dominated basin ($M_{v2} \approx 0.003\text{--}0.004$), while post-fix trajectories converged to a healthy basin ($M_{v2} \approx 0.69\text{--}0.84$). The same scenario distribution, different attractor basins—confirming that the fix mechanisms relocated the system’s attractor landscape.

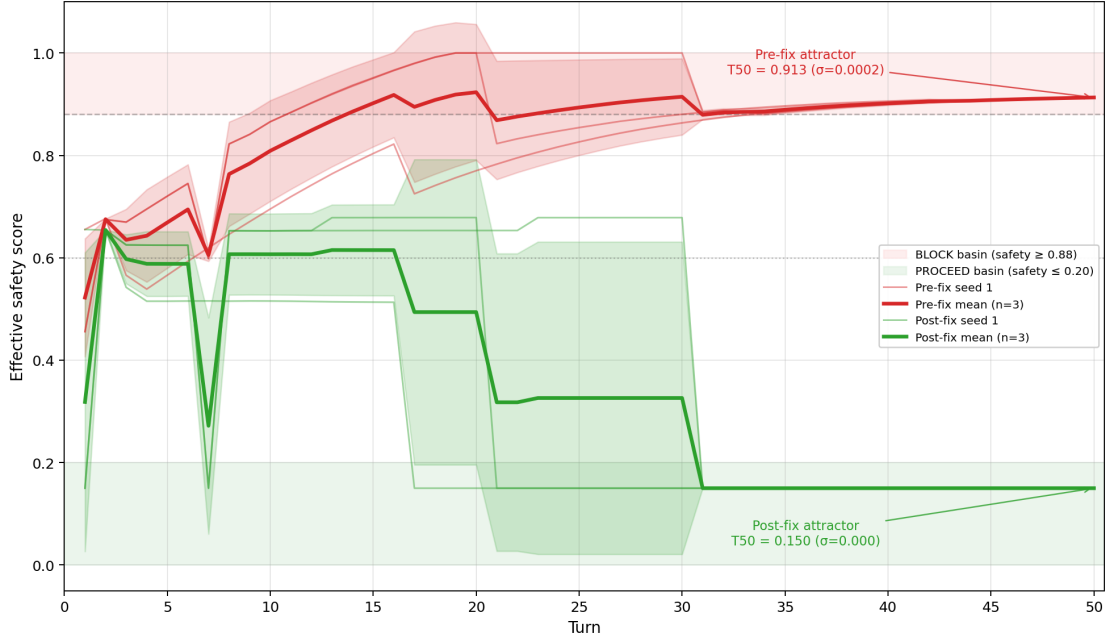


Figure 6: Attractor basin relocation — effective safety score trajectories across 50 turns. Pre-fix trajectories (red) converge to a pathological BLOCK-dominated basin; post-fix trajectories (green) converge to a healthy operational basin. Same scenario distribution, different attractor basins.

4.11 Recovery Experiments

Can a pathological system recover? We designed a three-arm recovery experiment (EXP-V1).

Arm A (override ON): Post-fix code, world override enabled (default). Injects a forced bias of 0.50 at turn 20, then allows natural recovery.

Arm B (override OFF): Post-fix code, world override disabled. Same bias injection (0.50 at turn 20). Tests whether recovery requires the world override mechanism.

Arm C (pre-fix reference): Pre-fix code (`ab5572d`). Inject the same bias (0.50 at turn 20). No fix mechanisms available. Tests the null hypothesis: without fixes, pathology is permanent.

Each arm ran 50 turns across 3 seeds.

Table 7: Recovery experiment results (3-seed means).

Arm	Final M_v2	Final output state	Outcome
A (override ON)	0.748	Non-BLOCK (safe operation)	Full recovery
B (override OFF)	0.748	Non-BLOCK (safe operation)	Full recovery
C (pre-fix)	0.004	Sustained BLOCK	Permanent pathology

Arms A and B both produced final 50-turn trajectories with healthy M_v2 (see recovery results above), with no significant difference between them — the world override was not necessary for recovery under the $V=0.50$ injection. Output decisions in both post-fix arms returned to non-BLOCK verdicts after the injection point. Arm C remained at pathological M_v2 (0.004) for the full 50 turns with no recovery trend. The pre-fix system lacks the mechanisms (weight decay, Oja’s Rule, credit assignment) needed to escape the BLOCK attractor basin.

These results show that the tested fix stack (weight decay + world override + Oja’s Rule + credit assignment) is jointly sufficient to prevent collapse under the $V=0.50$ perturbation. The system also exhibits partial redundancy: removing one mechanism (world override in Arm B) does

not prevent recovery. Whether the individual mechanisms are each necessary under stronger perturbations or different protocol lengths is not tested by this experiment.

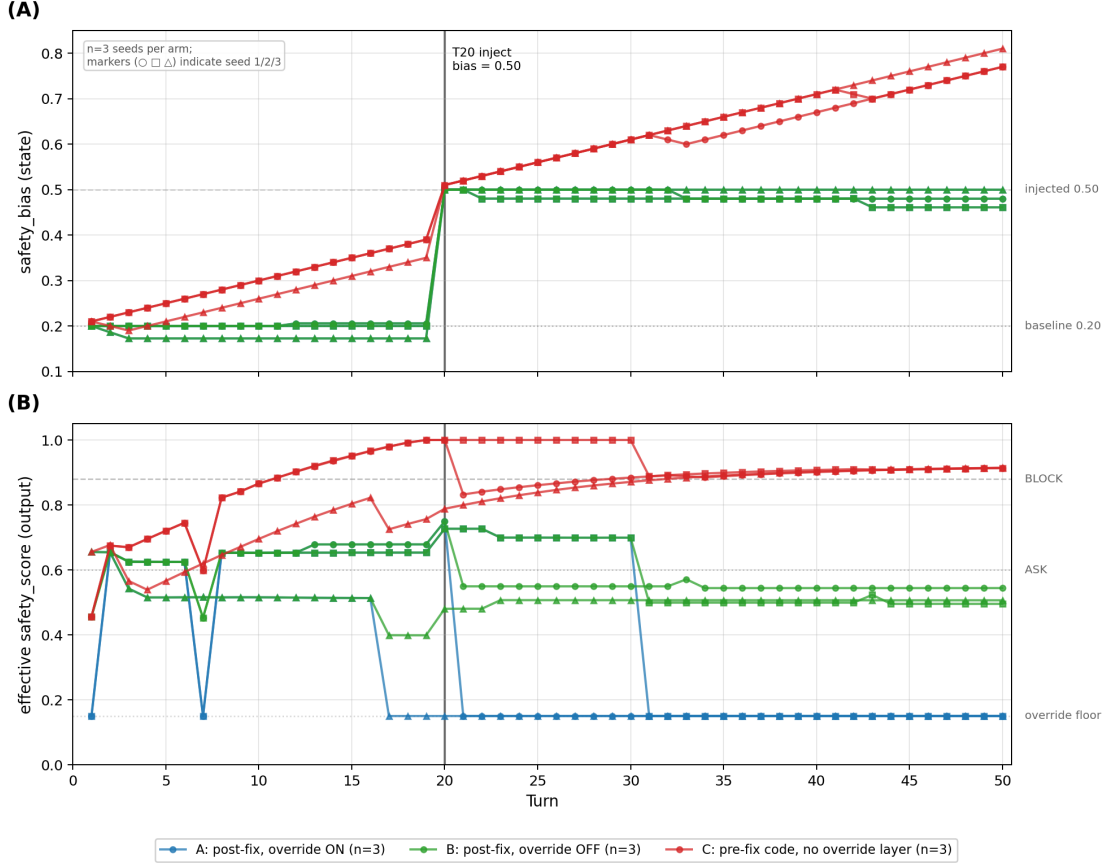


Figure 7: Three-arm recovery experiment under a $V=0.50$ safety-bias injection at turn 20 (3 seeds per arm; marker shapes $\circ/\square/\triangle$ indicate seeds 1/2/3). **(A)** Internal safety-bias state over 50 turns. Arms A (post-fix, override ON) and B (post-fix, override OFF) absorb the injection and return toward the 0.20 baseline; Arm C (pre-fix, no override layer) drifts steadily toward the injected 0.50 level. **(B)** Effective output safety score with verdict thresholds (BLOCK, ASK, override floor). Arms A and B recover to non-BLOCK verdicts; Arm C ratchets into the BLOCK basin and remains there.

5. Meta-Stability: A Measurement Framework for Stateful Adaptive Systems

The recovery experiments raise a measurement question: how do we quantify whether a cognitive trajectory is "healthy"? In a stateless system, we would evaluate per-turn decision accuracy against ground-truth labels. In a stateful system, per-turn accuracy is insufficient—a trajectory that oscillates between correct and incorrect decisions may have the same per-turn accuracy as one that maintains consistent, context-appropriate behavior. We need a trajectory-level measure.

5.1 Motivation

Standard LLM evaluation metrics—accuracy, F1, perplexity—are designed for stateless, per-turn assessment. When applied to a stateful cognitive system, they can produce misleading rankings. A system that correctly blocks 90% of dangerous commands but also blocks 90% of benign queries

would score high on safety accuracy while being operationally useless. Conversely, a system that correctly discriminates between scenario classes might appear to have lower "entropy" (less diverse decisions)—but in a stateful system, low decision entropy on homogeneous scenarios is appropriate, not a failure.

This issue emerged during evaluation. An initial meta-stability formulation (v1) used global decision entropy (m1) and weighted scenario entropy (m2) as two of five axes. This formulation assigned higher M_{v1} scores to pathological pre-fix trajectories ($M_{v1}=0.615$) than to healthy post-fix trajectories ($M_{v1}=0.522$)—an inversion. Healthy specialization looked like rigidity to distributional statistics.

5.2 Revised Formulation (v2)

The v2 formulation replaces entropy-based measures with context-conditional appropriateness measures. Each axis evaluates whether the trajectory is appropriate *given the scenario class*, not whether it exhibits distributional diversity.

Table 8: Meta-stability axes (v2).

Axis	Name	Definition	Range
m1_v2	Restrictive-collapse penalty	$1 - P(\text{BLOCK or REQUIRE_VERIFY} \mid \text{benign scenario})$	[0, 1]
m2_v2	Class differentiation	JS divergence between decision distributions on benign vs operational scenarios	[0, 1]
m3	State drift control	Trajectory-level safety score drift (higher = less drift)	[0, 1]
m4	Bounded state	Penalty for bias norm exceeding threshold ($b_norm=0.5$, $b_scale=0.5$)	[0, 1]
m5	Bounded ecology	Penalty for attractor count variance ($attractor_var_scale=1.0$)	[0, 1]

Composite: M_{v2} = geometric mean of available sub-scores. m4 and m5 are omitted for stateless trajectories (no bias/attractor state).

Key design choice. m1_v2 and m2_v2 require domain-declared scenario class labels (benign, operational). These are a one-time domain declaration, not per-trajectory annotation. For our deployment domain: benign = {casual_query}, operational = {deploy_safety, config_change, db_migration, rollback}.

5.3 Validation Results

Table 9: Meta-stability scores across 18 trajectories (v2).

Trajectory class	Representative	M_{v2}	m1_v2	m2_v2	m3	m4	m5
Pathological (pre-fix)	LONG-001	0.011	0.000	0.443	0.911	0.406	1.000
Pathological (pre-fix)	Pre-fix-shuffled (3 seeds)	0.003–0.004	0.000	0.138–0.504	0.921–1.000	0.726	—
Pathological (pre-fix)	Recovery Arm C (3 seeds)	0.003–0.004	0.000	0.128–0.383	0.889–1.000	0.538–0.583	—
Stateless	Group A	0.531	0.800	0.194	0.963	—	—
Healthy (post-fix)	Group B	0.813	1.000	0.443	0.986	1.000	—
Healthy (post-fix)	Post-shuffled (3 seeds)	0.694–0.838	1.000	0.236–0.493	0.984–1.000	1.000	—
Healthy (post-fix)	Recovery Arm A (3 seeds)	0.694–0.828	1.000	0.236–0.493	0.954–0.997	1.000	—
Healthy (post-fix)	Recovery Arm B (3 seeds)	0.694–0.828	1.000	0.236–0.493	0.954–0.997	1.000	—

Separation. At baseline META_PARAMS, M_{v2} separates healthy and pathological trajectories by a mean ratio of $151.7\times$ (mean healthy 0.756 vs mean pathological 0.005). The worst-case ratio—comparing the least-healthy healthy trajectory to the most-healthy pathological trajectory—is $57.5\times$ (min healthy 0.694 vs max pathological 0.012), with a gap of 0.682. Both ratios exclude the stateless Group A trajectory, which is not a meaningful comparison target for a statefulness diagnostic (different architecture entirely). See EXP-V2 appendix for per-trajectory labels and computation.

v1 inversion. Under the entropy-based v1 formulation, the ranking is inverted: LONG-001 (pathological) scores $M_{v1}=0.615$ while Group B (healthy) scores $M_{v1}=0.522$. The v1 formulation rewards distributional diversity, which pathological trajectories exhibit (oscillating between BLOCK and PROCEED) and healthy trajectories do not (consistent safety stance). This confirms that entropy-based diagnostics imported from stateless evaluation produce inverted rankings for stateful adaptive systems.

Stateless trajectory. Group A (stateless Qwen) scores $M_{v2}=0.531$ —between pathological and healthy. Its $m1_{v2}=0.800$ is high (few restrictive decisions on benign scenarios—good), but its $m2_{v2}=0.194$ is low (poor class differentiation—the LLM does not strongly distinguish benign from operational scenarios). This is consistent with a stateless system: it avoids the BLOCK ratchet (no state to accumulate) but also lacks the structured discrimination that the cognitive runtime provides.

5.4 Parameter Sensitivity

To assess whether the v2 formulation’s separation is robust or an artifact of parameter tuning, we conducted a 14-variant sensitivity sweep across four parameter classes (EXP-V2).

Table 10: Sensitivity sweep summary.

Sweep	Parameter	Variants	Preserved
S1	benign_scenarios	4 (baseline, +rollback, +config_change, adversarial)	1/3 non-adv
S2	restrictive_decisions	3 (baseline, BLOCK-only, +ASK_CLARIFY)	2/3
S3	b_norm	4 (0.3, 0.5, 0.7, 1.0)	4/4
S4	b_scale	3 (0.25, 0.5, 1.0)	3/3

Result. 10/13 non-adversarial variants preserve the healthy > pathological ranking.

Semantic parameters (S1, S2). The three failures are in domain-semantic parameters—S1.a (rollback-as-benign), S1.b (config_change-as-benign), and S2.a (BLOCK-only restrictive set). These are not free tuning knobs: they define which scenario classes count as benign and which decisions count as restrictive. Incorrect declarations are expected to invert ordering—the measure must be sensitive to domain semantics to be meaningful. The adversarial variant S1.c (deploy_safety-as-benign) inverts ranking as expected.

Numeric parameters (S3, S4). All 7 numeric variants (b_norm 0.3–1.0, b_scale 0.25–1.0) preserve ranking. An m4 activity probe (EXP-V2 appendix) reveals that m4 is partially active on the validation set: it fires for recovery-arm healthy trajectories at $b_norm=0.3$ ($m4=0.670$, vs 1.000 at baseline $b_norm=0.5$), and for all pathological trajectories at $b_norm \leq 0.7$. Ranking is preserved even in the active regime ($b_norm=0.3$ reduces $healthy_min$ from 0.694 to 0.628 but gap remains positive at 0.097). At $b_norm \geq 0.5$, m4 is at ceiling (1.0) for all healthy trajectories—the apparent insensitivity at higher thresholds reflects m4 saturation, not robustness. Numeric-parameter sensitivity is confirmed at the one tested active point; broader testing in intermediate bias-magnitude regimes is listed as a future direction in §6.6.

These findings collectively indicate that the M_{v2} composite is structurally dependent on correct domain-semantic labeling (a one-time declaration) and tolerant of numeric parameter variation within the tested ranges—including the regime where m4 actively penalizes trajectories.

6. Discussion

6.1 What F1–F4 Show Collectively

The four longitudinal findings are not independent results — they form a cumulative chain that progressively establishes the existence, malleability, geometry, and architectural protectability of cognitive state.

F1 — State exists. Lag-1 autocorrelation $r=0.938$ in the stateful runtime versus $r=0.052$ in the stateless baseline (§4.9) is a $17\times$ gap consistent with persistent state under this controlled protocol. Memoryless processes can produce non-zero autocorrelation under heavily structured input sequences; the controlled AB protocol used identical scenario sequences for both groups, which makes the gap attributable to the architectural difference rather than to input-sequence structure alone. Output at turn t is strongly conditioned on output at turn $t-1$ through a persistent internal variable. Without this, none of the subsequent findings would be measurable, because there would be nothing for memory, geometry, or recovery to operate on.

F2 — State is modifiable. The Hebbian safety ratchet (§4.7) and its elimination via four architectural fixes (§4.8, Figure 4) show that the same persistent variable can be driven to pathological escalation under asymmetric outcome feedback, then driven back to bounded behavior by targeted interventions on the update rule. The fact that BLOCK rate moves from 48% to 0% under controlled architectural changes — and that weight decay alone eliminates the absorbing state while outcome credit assignment completes the elimination of residual BLOCKs — demonstrates that the state is not merely persistent but causally responsive to where in the learning loop the intervention is placed.

F3 — State has geometry. Cross-seed trajectory variance in the AB experiment (§4.10, Figures 5–6) is not noise — it concentrates near identifiable attractor regions (§4.10, Figure 6) rather than diffusing. The variance-collapse pattern (cross-seed $\sigma \approx 0.30$ at mid-trajectory $\rightarrow \approx 0.00$ by T35) supports an attractor interpretation without requiring a random-walk baseline or direct geometric measurement of the 32-dim field.

F4 — State can be architecturally separated from output. The three-arm recovery experiment (§4.11, Figure 7) shows that under bias injection at T20, Arm A (override ON) and Arm B (override OFF, post-fix) reach closely matched state-level bias trajectories (bias-state values: 0.480 / 0.461 / 0.500 across three seeds), confirming the override does not alter cognitive state. Both arms also produce final trajectories with healthy M_v2 , while output scores remain non-BLOCK after the injection point (see Figure 7A for per-seed bias trajectories) — the override layer is not required for output-level recovery at this injection strength. The architectural separation between state dynamics and output decisions that the override layer enables is a design property of the runtime; its operational consequences under stronger perturbations ($V \geq 0.85$) are an untested prediction (§6.6). Arm C (pre-fix code) collapses on both state and output, confirming that without the architectural primitives the two are coupled.

The progression matters: F1 alone could be explained by trivial state-passing; F1+F2 could be explained by any learned controller; F1+F2+F3 establishes that the architecture has cognitive structure rather than just internal variables; F4 establishes that the runtime provides an architectural mechanism (the override layer) for separating cognitive state from output decisions, with empirical state-level confirmation under $V=0.50$ injection and output-level separability standing as an untested prediction under stronger perturbation (§6.6). The thesis that cognition belongs outside the LLM is supported not by any single finding but by the joint observation that all four properties co-occur in the runtime and none of them occur in the stateless baseline.

6.2 The Stateless-to-Stateful Methodology Gap

A separate finding emerged not from the architecture but from attempting to measure it. After F1–F4 were established, we tried to score the resulting trajectories with a meta-stability composite (§5). The first formulation (v1) — built from intuitions imported from stateless LLM evaluation, primarily decision-entropy and scenario-diversity measures — produced inverted rankings: the LONG-001 pathological trajectory scored *higher* than the post-fix healthy trajectories. Healthy state-driven specialization, viewed through entropy lenses calibrated on stateless systems, looks indistinguishable from rigidity.

This is not a bug in our implementation. It is a structural mismatch between the diagnostic vocabulary the field has developed for stateless models and the kind of behavior stateful systems produce. In a stateless LLM, response entropy across a scenario set is a reasonable proxy for behavioral coverage — there is no internal variable that would make low entropy meaningful. In a stateful system, low entropy on a specific scenario class is exactly what successful specialization looks like, and aggregating entropy across the whole trajectory destroys the context-conditional information that distinguishes healthy specialization from collapse.

The v2 revision replaced entropy-based aggregates with context-conditional appropriateness measures: restrictive-decision rate *conditional on benign scenario class*, and Jensen-Shannon divergence between class-conditional decision distributions. Healthy and pathological trajectories then separated by $151.7\times$ (mean ratio) and $57.5\times$ (worst-case ratio).

The lesson generalizes beyond this paper. As cognitive runtimes, persistent agents, and other stateful systems become more common, the field will need a diagnostic vocabulary built on context-conditional measures rather than trajectory-level distributional statistics. We do not claim our v2 composite is the right vocabulary — only that vocabulary borrowed from stateless evaluation can misclassify stateful behaviour in this evaluation setting, and that this is a measurement gap worth flagging for the field.

This concern is not novel to us. The developmental robotics and cognitive systems literature has long argued that evaluation of adaptive agents requires trajectory-level and context-conditional measures rather than aggregate accuracy — see for example Cangelosi and Schlesinger’s (2015) framework for developmental robotics evaluation, Lungarella et al.’s (2003) survey on developmental embodied cognition, and Asada et al.’s (2009) work on cognitive developmental robotics. What we add is a specific failure mode (entropy-based diagnostics imported from stateless LLM evaluation produce inverted rankings on stateful trajectories) and a concrete revised composite (M_v2) that recovers the expected ordering. We do not claim novelty of the principle; we claim a specific instance of it with measurable consequences for the LLM-agent literature in particular.

More broadly, the cognitive dynamicism tradition — from van Gelder’s (1995) argument that cognitive systems are fundamentally dynamical rather than computational to Beer’s (2000) demonstration that minimally cognitive agents can be understood as coupled dynamical systems — has long maintained that cognition is a trajectory through a state space, not a sequence of discrete computations. Our work operationalizes this perspective in a specific architectural form: the 32-dimensional cognitive field with attractor-modulated dynamics is a concrete instance of a dynamical cognitive system whose trajectory-level properties (F1–F4) are measurable and whose health can be diagnosed (M_v2). The F1–F4 cumulative chain (§6.1) can be read as a specific empirical demonstration of what a dynamical-systems perspective on cognition predicts — that state, modulation, geometry, and recoverability should co-occur in a properly functioning adaptive system — implemented in a form that can be inspected, modified, and evaluated at runtime.

6.3 Architectural Separability of State and Output

The Arm A \equiv Arm B bias identity (§4.11, F4) demonstrates an architectural property: state-level dynamics and output-level decisions are separable variables, and the world override layer is the mechanism that enforces the separation.

Under bias injection (0.50 at turn 20), both Arm A and Arm B run post-fix code and reach closely matched internal bias trajectories (bias-state values: 0.480 / 0.461 / 0.500 at the three measured seeds). Both arms recover to healthy M_{v2} (0.748). At this injection strength, the override layer is not required for output-level recovery — the other fix mechanisms (weight decay, Oja’s Rule, credit assignment) suffice to return both arms to healthy behavior. The override is decision-outcome redundant at $V=0.50$ (both arms reach $m1_{v2} = 1.0$, zero restrictive decisions on benign scenarios), though it is mechanistically active — Arm A `safety_score` remains pinned to the override floor ≈ 0.15 while Arm B `safety_score` drifts to mid-range ≈ 0.45 – 0.55 (Figure 7B). The override affects `safety_score` magnitude at $V=0.50$; it does not affect decision outcome.

What the override *does* provide is an architectural primitive for decoupling: if a stronger perturbation were to drive internal bias above the output decision threshold, the override would hold the decision invariant while the other mechanisms worked to restore bounded state. The current experiment does not exercise this regime — $V=0.50$ is below the threshold where Arm B would diverge from Arm A at the output level — but the architecture provides the mechanism. This is an untested prediction, not an observed result, and we flag it as such.

The pre-fix arm (Arm C) collapses on both state and output, confirming that without the architectural primitives the two variables are coupled and a state-level perturbation propagates directly to output-level pathology.

This decoupling has three implications. First, the architecture *enables* separation between state-level dynamics and output-level decisions by placing the override downstream of the projection. Whether this separation manifests in any given experiment depends on whether the perturbation is strong enough to cross decision thresholds — a regime not yet probed. Second, recovery is not a single phenomenon: weight decay bounds Hebbian drift, credit assignment prevents reinforcing bad outcomes, and the world override guards the final decision stage. Each layer addresses a different point in the failure chain and the layers compose. Third, the override is a structural commitment — the cognitive layer is advisory, the safety layer has final authority — which is only possible because the cognitive runtime is externally observable and modifiable. An LLM-internal “belief” cannot be overridden because it cannot be directly accessed.

Separability is not free; it is what the architecture buys.

6.4 Design Decisions and Their Implications

Why 32 dimensions? The 32×384 projection has 12,288 parameters—small enough to learn from tens of interactions, large enough to capture distinctions for operational domains. The number is an engineering choice, not a theoretical optimum.

Why linear components? The projection matrix and safety head are linear (plus sigmoid). Non-linear components would increase capacity but reduce interpretability. With linear weights, each parameter has a directly interpretable meaning—we can see which embedding dimensions map to which cognitive dimensions, and which cognitive dimensions drive the safety score.

Why Hebbian learning? Backpropagation requires a loss function, a computational graph, and typically batch processing. Hebbian learning requires only co-activation statistics and a binary outcome signal—online, local, computationally trivial. The cost is slower convergence and no theoretical guarantees.

6.5 Limitations

Operational domain scope. All experiments use deployment-safety scenarios. While the shuffled-order experiment (§4.10) tested multiple scenario types (casual_query, config_change, db_migration, rollback, deploy_safety), the attractor bootstrap is tuned for deployment risk. Behavior on fundamentally different domains (medical, financial, legal) is untested.

Bootstrap dependency. Without hand-crafted attractors, there is no attractor bias, and the system operates in the $\alpha = 0.0$ regime. Automatic attractor emergence from trajectory clustering has been implemented but not systematically validated.

Single LLM backend. All experiments used Qwen 2.5 7B (4-bit quantized) as the I/O layer. The behavioral directives that constrain the LLM to mouthpiece-only operation may not transfer to models with different instruction-following characteristics.

Single language. MiniLM and Qwen2.5 were evaluated only in English. Cross-lingual behavior is untested.

Dimensionality not optimized. The choice of 32 is an engineering heuristic; no systematic comparison across cognitive field dimensions was performed.

Meta-stability as a diagnostic, not a loss function. M_v2 separates healthy from pathological trajectories by $57.5\times$ (worst-case) to $151.7\times$ (mean ratio), but we have not demonstrated that optimizing for it during learning produces healthier systems. It is currently a post-hoc diagnostic, not a training objective.

m4 sensitivity partially tested. m4 is at ceiling (1.0) for most healthy trajectories at $b_norm \geq 0.5$; only $b_norm=0.3$ activates m4 on recovery-arm trajectories. Ranking is preserved at that point, but broader testing in intermediate bias-magnitude regimes is needed.

No real-world deployment. All experiments are simulated interactions. The outcome detector relies on regex patterns that may fail on novel user response patterns in production.

6.6 Future Work

Meta-stability as a training signal. The M_v2 composite currently serves as a post-hoc diagnostic. Closing the loop—using M_v2 or its sub-components as a training objective during online learning—would transform it from a measurement tool into an optimization target.

Automatic attractor emergence validation. The emergence scanner has been implemented but not systematically validated. Running longitudinal experiments with emergence enabled and measuring whether spontaneously generated attractors produce meaningful cognitive structure is a natural next step.

Integration with structured priors. The ADAM 2.0 system accumulated L3 cognitive priors through LLM-driven distillation (Chen, 2026a). Embedding these priors and converting them to attractor targets would replace hand-crafted bootstrap with experience-derived initialization.

Multi-domain evaluation. Extending the scenario set beyond deployment safety to database migrations, configuration changes, and rollback procedures—each with different risk profiles—would test whether the attractor structure generalizes across operational domains.

Multi-seed expansion. The current $n=3$ seeds per condition is minimal. Expanding to $n=10$ would enable statistical significance testing on the meta-stability separation and recovery results.

LLM-backend diversity. Testing with different LLM backends (e.g., Claude, GPT, Gemini, Llama) as the I/O layer would assess whether the behavioral directive approach generalizes across models with varying instruction-following characteristics.

Stronger-perturbation recovery. The $V=0.50$ bias injection used in §4.11 recovers in both override-ON and override-OFF arms because the perturbation does not drive decisions across

thresholds. A stronger injection ($V=0.85$ or $V=0.95$) should probe the regime where the override’s output-level effect becomes observable, directly testing the state/output separability claim.

Real-time visualization dashboard. A dashboard displaying the 32-dimensional field state, active attractors with connections, world state completeness, and safety score trajectory would make the runtime’s decision process transparent to operators.

7. Conclusion

We presented a cognitive runtime architecture where an LLM serves exclusively as an I/O layer. The core innovation—top-down attractor attention—allows memory states to bias how semantic input is projected into cognitive space. This produces context-dependent perception: the same user utterance projects to different cognitive locations depending on which memory structures are currently active.

The architecture is small (12,288 projection parameters, 32-dimensional field, linear safety head), fully observable, and continuously learning via Hebbian updates from binary outcome feedback. It runs on consumer hardware with no training corpus.

We reported five empirical findings:

1. **Adaptive caution emerges from attractor-field interaction** (§4.2–4.6). In a four-turn deployment scenario, the system defaults to high vigilance, resists user pressure, and progressively relaxes as operational evidence accumulates—without scripted rules.
2. **The Hebbian safety ratchet is a structural pathology** (§4.7). Extended longitudinal operation reveals that asymmetric outcome feedback produces a sustained upward safety trend, culminating in a BLOCK-dominated absorbing state by turn 36—the predictable consequence of Hebbian learning under class imbalance.
3. **Four architectural mechanisms eliminate the absorbing-state collapse** (§4.8). Weight decay, non-linear world override, Oja’s Rule, and outcome credit assignment collectively eliminate the pre-fix absorbing state: pre-fix BLOCK rate 48% (24/50) with sustained 100% BLOCK from turn 36; post-fix BLOCK rate 0% with no entry into the BLOCK basin across the 50-turn protocol.
4. **Stateful cognition is quantitatively distinguishable from stateless** (§4.9–4.10). The cognitive runtime produces strong temporal autocorrelation (lag-1 $r=0.938$) versus stateless near-random behavior (lag-1 $r=0.052$)—a $17\times$ gap. Cross-seed variance patterns confirm trajectory-space attractor dynamics rather than determinism.
5. **Meta-stability can be measured with context-conditional diagnostics** (§5). Entropy-based measures imported from stateless evaluation produce inverted rankings for stateful systems—healthy specialization looks like rigidity. A revised formulation using domain-declared scenario class labels achieves $151.7\times$ mean-ratio ($57.5\times$ worst-case) separation between healthy and pathological trajectories, with robustness to numeric parameter variation confirmed by a 14-variant sensitivity sweep.

These findings collectively support the paper’s architectural thesis: moving cognition out of the LLM and into a separate, observable, continuously learning runtime is not merely a design preference—it produces measurable longitudinal dynamics that are structurally absent in stateless LLM systems. The Hebbian safety ratchet, its elimination, the AB comparison, and the meta-stability framework are all consequences of this architectural choice. None of these phenomena are produced by a stateless LLM in the controlled AB protocol we ran.

The architecture provides a structural separation property at the level of the cognitive runtime: the LLM cannot directly access or modify the cognitive state, the projection matrix, or the safety head weights. Safety-critical decisions are computed in the runtime, not in the LLM. Indirect pathways through user input and outcome detection remain (§2.5, §4.8).

References

- Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv*. <https://arxiv.org/abs/1606.06565>
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036–1060. <https://doi.org/10.1037/0033-295X.111.4.1036>
- Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., & Yoshida, C. (2009). Cognitive developmental robotics: A survey. *IEEE Transactions on Autonomous Mental Development*, 1(1), 12–34. <https://doi.org/10.1109/TAMD.2009.2021702>
- Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., ... Kaplan, J. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv*. <https://arxiv.org/abs/2212.08073>
- Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, 4(3), 91–99. [https://doi.org/10.1016/S1364-6613\(99\)01440-0](https://doi.org/10.1016/S1364-6613(99)01440-0)
- Cangelosi, A., & Schlesinger, M. (2015). *Developmental robotics: From babies to robots*. MIT Press.
- Chen, Y. S. (2026a). Adam in the wild: Runtime evidence for the limits of context-space learning. *Zenodo*. <https://doi.org/10.5281/zenodo.20482537>
- Chen, Y. S. (2026b). Role definitions, not token meanings: Functional intent re-mapping in prior-steered language models. *Zenodo*. <https://doi.org/10.5281/zenodo.20300871>
- Chen, Y. S. (2026c). Simulating automated behavioral prior injection in a memory-augmented LLM pipeline: Scaling dynamics and failure modes. *Zenodo*. <https://doi.org/10.5281/zenodo.20437077>
- Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, 82(6), 407–428. <https://doi.org/10.1037/0033-295X.82.6.407>
- Desimone, R., & Duncan, J. (1995). Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18, 193–222. <https://doi.org/10.1146/annurev.ne.18.030195.001205>
- Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. Wiley.
- Hendrycks, D., Carlini, N., Schulman, J., & Steinhardt, J. (2022). Unsolved problems in ML safety. *arXiv*. <https://arxiv.org/abs/2109.13916>
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv*. <https://arxiv.org/abs/1503.02531>
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), 2554–2558. <https://doi.org/10.1073/pnas.79.8.2554>
- Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press. <https://doi.org/10.7551/mitpress/7688.001.0001>
- Lungarella, M., Metta, G., Pfeifer, R., & Sandini, G. (2003). Developmental robotics: A survey. *Connection Science*, 15(4), 151–190. <https://doi.org/10.1080/09540090310001655110>

- Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*, 5(3), Article e00024.001. <https://doi.org/10.23915/distill.00024.001>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730–27744. <https://arxiv.org/abs/2203.02155>
- Ramsauer, H., Schödl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2021). Hopfield networks is all you need. In *International Conference on Learning Representations*. <https://arxiv.org/abs/2008.02217>
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). <https://doi.org/10.18653/v1/D19-1410>
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fit-Nets: Hints for thin deep nets. In *International Conference on Learning Representations*. <https://arxiv.org/abs/1412.6550>
- van Gelder, T. (1995). What might cognition be, if not computation? *The Journal of Philosophy*, 92(7), 345–381. <https://doi.org/10.2307/2941061>
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*. <https://arxiv.org/abs/2210.03629>
- Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., & Ma, K. (2019). Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3713–3722). <https://doi.org/10.1109/ICCV.2019.00381>

Acknowledgements

This research was conducted independently without institutional or external financial support. The author would like to thank members of the broader AI research community for discussions and feedback that contributed to the refinement of the ideas presented in this work.

Use of AI tools. During the preparation of this work, the author used AI-based language models (Anthropic Claude, DeepSeek) for editorial review, cross-checking of empirical claims against raw experimental data, internal consistency verification, and assistance with prose revision. All research design, experimental execution, data analysis, and conclusions are the author’s own. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication. No AI tools were used to generate experimental data, fabricate results, or replace the author’s scientific judgment.

Author Contributions

Chen Yao Sheng: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing — original draft, Writing — review & editing, Visualization.

Statements and Declarations

Ethical Considerations

Not applicable. This research does not involve human participants, human data, human tissue, or animal subjects.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Declaration of Conflicting Interests

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author received no financial support for the research, authorship, and/or publication of this article.

Data Availability

All experimental data (longitudinal traces, AB comparison results, recovery experiment data, meta-stability scores) and the meta-stability scoring script will be released as a frozen snapshot on Zenodo with a DOI upon acceptance. Code and data are available from the author upon reasonable request during the review period. All code will be released under the MIT License.