Unified Governance and Memory Equations for Autonomous AI Agent Systems: Confidence-Gated Execution, Ebbinghaus-Anchored Retention, and Cryptographic Enforcement in Production Deployments

Billy Whited

Independent Researcher, Atlas UX Research

ORCID: [0000-0000-0000-0000]

Abstract

Autonomous AI agent systems operating without continuous human oversight require formal guarantees that unsafe actions cannot execute and that accumulated knowledge persists across sessions without unbounded growth. We present a unified mathematical framework combining three interlocking equations: (1) a governance-gated execution equation $\hat{\tau}_{i,t} = f_\theta(g, s_t, A_t, \tau_i) \cdot \mathbb{1}[c \geq \gamma(r)]$ that gates every agent output through a confidence threshold conditioned on action risk tier, (2) an Ebbinghaus-adapted memory retention equation $M(I, t) = P_{\text{kdr}}(I) + (1 - P_{\text{kdr}}(I)) \cdot e^{-t/S(I)} \cdot [w_1 \cdot \text{freq}(I) + w_2 \cdot \text{impact}(I) + w_3 \cdot \text{conn}(I)] \cdot (1 + \beta \cdot P_{\text{ltm}}(I))$ that combines exponential forgetting with anchored state persistence through Key Decision Records (KDRs), and (3) a risk-adjusted threshold function $\gamma(r) = \gamma_{\text{base}} + \alpha \cdot r$ mapping discrete risk tiers to minimum confidence requirements. We prove that the confidence-gated indicator function $\mathbb{1}[c \geq \gamma(r)]$ provides a formal safety guarantee: no action with confidence below the risk-adjusted threshold can reach external systems, reducing the space of possible unsafe executions to the false-positive region of the confidence estimator. We further prove that KDR anchoring ($P_{\text{kdr}} = 1$ for anchored information) defeats the Ebbinghaus forgetting curve for decision-critical state, achieving flat retention independent of time. We validate both equations against production deployment data from Atlas UX, a 33-agent autonomous platform with cryptographic audit enforcement. We compare our framework to Aegis cryptographic runtime governance, SAGE self-evolving reflective memory, AgentGuardian learned access control, and cuGenOpt's Adaptive Operator Selection, showing that our unified approach subsumes the safety guarantees of each while adding production-validated memory persistence.

Keywords: autonomous agent safety, governance equations, Ebbinghaus forgetting curve, confidence gating, cryptographic audit, multi-agent systems, memory-augmented agents, formal verification

JEL Classification: C63, D81, L86, O33

---

1. Introduction

1.1 The Autonomous Agent Safety Problem

The deployment of autonomous AI agent systems in production environments has accelerated dramatically since 2024, with frameworks such as ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023), and Generative Agents (Park et al., 2023) demonstrating that large language models can be composed with tool-use capabilities, memory systems, and planning mechanisms to form agents that operate with increasing independence. The 2025 AI Agent Index (Fang et al., 2026) documents the rapid proliferation of deployed agentic systems while simultaneously identifying a critical finding: most production agents lack formal safety guarantees. The gap between what agents can do and what agents can be proven to do safely represents the central challenge of the field.

This challenge is not merely academic. An autonomous agent managing business operations — booking appointments, processing payments, sending communications, modifying databases — operates in a regime where incorrect actions have material consequences. A miscalibrated confidence score that allows an unauthorized financial transaction, a memory failure that causes the agent to forget a customer's cancellation request, or an audit trail gap that prevents forensic reconstruction of an incident all represent failure modes that no amount of prompt engineering can reliably prevent.

The fundamental insight motivating this work is that agent safety and agent memory are not independent problems. An agent that cannot reliably remember its governance constraints is an agent whose safety guarantees degrade over time. Conversely, an agent whose memory grows without bound will eventually drown in irrelevant context, degrading the quality of its governance decisions. The two problems are coupled, and any framework that addresses them independently will produce solutions that are incomplete at best and contradictory at worst.

1.2 Why Advisory Safety Does Not Scale

Current approaches to AI agent safety rely on three mechanisms, all of which fail under fully autonomous operation.

Prompt-based guardrails instruct the model to adhere to safety constraints through system prompts. Statements such as "never reveal personal information" or "always request approval

before spending money" are effective for supervised chat applications but are trivially bypassed through prompt injection, context overflow, or multi-turn manipulation (Mazzocchetti, 2026). An autonomous agent running 24/7 without human oversight cannot rely on prompt-level safety because the enforcement mechanism — the model's tendency to follow instructions — is probabilistic, not deterministic.

Output filtering applies post-hoc scans to agent outputs, checking for policy violations such as toxicity, PII leakage, or unauthorized actions. This approach catches known violation patterns but produces false negatives on novel attacks, introduces latency that degrades real-time performance, and — critically — operates after the agent has already decided to violate policy. Output filters address symptoms rather than causes.

Human-in-the-loop (HIL) escalation routes high-risk decisions to human reviewers. This works when agent throughput is low and human reviewers are available. At scale — thousands of decisions per hour, across time zones, during off-hours — HIL becomes either a bottleneck that blocks agent operation or a rubber-stamp process that defeats its own purpose. Neither outcome is acceptable for production systems.

None of these approaches provide verifiable safety guarantees. There is no mechanism by which an auditor, regulator, or user can independently verify that the agent's safety constraints were actually enforced for any given action. The agent's implicit claim "I followed my safety rules" is unfalsifiable under these paradigms.

1.3 The Memory-Governance Coupling

The coupling between governance and memory manifests in three directions.

First, governance decisions require memory context. When an agent evaluates whether a proposed action exceeds its risk threshold, the quality of that evaluation depends on the agent's memory of similar past actions, their outcomes, and any governance constraints that applied. An agent with degraded memory makes worse governance decisions — not because its reasoning capability has diminished, but because it lacks the contextual information needed for accurate risk assessment.

Second, memory management itself requires governance. Decisions about what to remember, what to forget, and what to archive are consequential. An agent that forgets a critical safety constraint because it was not retrieved frequently enough has experienced a governance failure through a memory mechanism. The memory system must itself be subject to governance policies that prevent safety-critical information from decaying.

Third, the audit trail — the cryptographic record that enables post-hoc verification of governance decisions — is itself a memory system. If the audit trail is incomplete, corrupted, or inaccessible, the governance guarantees it supports become unverifiable. The integrity of governance depends on the integrity of memory, and vice versa.

This coupling motivates the unified framework we present: governance and memory equations that are designed together, share formal properties, and reinforce each other's guarantees.

1.4 Contributions

This paper makes four contributions:

1. **A formal governance equation** $\hat{\tau}_{i,t} = f_\theta(g, s_t, A_t, \tau_i) \cdot \mathbb{1}[c \geq \gamma(r)]$ with a proven safety guarantee that no sub-threshold action reaches external systems, and a risk-adjusted threshold function $\gamma(r) = \gamma_{\text{base}} + \alpha \cdot r$ that maps discrete risk tiers to minimum confidence requirements.
2. **A KDR-anchored memory equation** $M(I, t) = P_{\text{kdr}}(I) + (1 - P_{\text{kdr}}(I)) \cdot e^{-t/S(I)} \cdot [w_1 \cdot \text{freq}(I) + w_2 \cdot \text{impact}(I) + w_3 \cdot \text{conn}(I)] \cdot (1 + \beta \cdot P_{\text{ltm}}(I))$ that extends the Ebbinghaus forgetting curve with persistent anchoring, proving that anchored information achieves flat retention independent of time.
3. **A formal coupling analysis** showing why governance and memory must be co-designed, with proofs that memory degradation produces governance failures and that governance constraints must protect memory integrity.
4. **Production validation** across 33 named agents on Atlas UX, demonstrating that the equations describe real system behavior with cryptographic audit enforcement, hash-chained integrity verification, and automated self-healing.

2. Related Work

2.1 Constitutional AI and Prompt-Level Safety

The constitutional AI paradigm, introduced by Anthropic (Bai et al., 2022), embeds ethical principles into the AI system's training process and system prompts, guiding behavior through learned preferences rather than hard constraints. While effective at steering model behavior in supervised settings, constitutional AI operates at the level of tendency rather than guarantee. A model trained to be helpful, harmless, and honest will generally behave accordingly, but the "generally" qualification is precisely the problem for autonomous agents where a single safety violation can have irreversible consequences. The distinction between "the model tends to follow this rule" and "the model is mathematically prevented from violating this rule" is the distinction between advisory and enforceable safety — the same distinction that separates building codes from building materials.

2.2 Aegis: Cryptographic Runtime Governance

Mazzocchetti (2026) introduced the Aegis Architecture for Verifiable Policy Enforcement (arXiv:2603.16938), representing a paradigm shift from constitutional AI as prompt engineering to constitutional AI as systems engineering. Aegis binds each governed agent to an Immutable Ethics Policy Layer (IEPL) at system genesis, enforces all external emissions through an Ethics Verification Agent (EVA) using formal predicate checking, and produces cryptographic proof artifacts at every step through an Enforcement Kernel Module (EKM) and Immutable Logging Kernel (ILK).

Aegis achieves zero false negatives on hard constraints (every defined policy violation is caught before reaching external systems) with a 2.1% false positive rate and 238ms median proof verification latency. The architecture demonstrates that agent safety can be moved from the domain of alignment research into the domain of verifiable computing, where guarantees are mathematical rather than probabilistic.

Our governance equation formalizes the relationship between Aegis's architectural components and the mathematical conditions under which safety guarantees hold. Specifically, we show that the IEPL maps to the governance policy g, EVA maps to the confidence evaluator c, the EKM maps to the indicator function $\mathbb{1}[c \geq \gamma(r)]$, and the ILK maps to the hash-chained audit trail.

2.3 SAGE: Self-Evolving Agents with Reflective Memory

Liang, He, Xia et al. (2024) introduced SAGE (Self-evolving Agents with reflective and memory-augmented abilities, arXiv:2409.00872), proposing agents that improve through iterative feedback loops, reflective self-analysis, and biologically-inspired memory management. SAGE organizes agent memory into short-term memory (STM) for session context and long-term memory (LTM) for persistent knowledge, with a reflection mechanism bridging the two through compressed, actionable lesson extraction.

The most novel aspect of SAGE's memory system is its application of the Ebbinghaus forgetting curve to manage LTM contents. Each entry in LTM has an associated retention score $R = e^{-t/S}$ that decays over time, where t is time since encoding and S is memory strength. Memories that are accessed frequently have their retention scores refreshed; memories that are never accessed decay toward zero and are pruned. This biological model provides a principled mechanism for managing unbounded memory growth while preserving demonstrably useful information.

SAGE evaluates across eight environments (code, OS, database, Q&A, knowledge graph, Mind2Web, ALFWorld, web shopping), demonstrating that the self-evolution pattern generalizes across domains. Our memory equation extends SAGE's Ebbinghaus model with KDR anchoring, proving that certain categories of information — specifically, decision-critical state records — should be exempt from exponential decay.

2.4 AgentGuardian: Learned Access Control Policies

Liu et al. (2026) introduced AgentGuardian (arXiv:2601.10440), which learns access control policies to govern AI agent behavior by automatically generating permission rules based on observed agent actions and declared capabilities. AgentGuardian complements our constitutional approach with learned behavioral constraints, addressing the case where governance policies must be discovered from operational data rather than specified a priori.

Our framework incorporates AgentGuardian's insight that some governance constraints should be learned rather than prescribed, while maintaining that safety-critical constraints must be formally specified and cryptographically enforced. The learned constraints inform the confidence estimator c while the formal constraints define the threshold function $\gamma(r)$.

2.5 Governance-as-a-Service

Li et al. (2025) proposed Governance-as-a-Service (GaaS, arXiv:2508.18765), a multi-agent framework for AI system compliance and policy enforcement that externalizes governance from

the agent itself into a service layer. This architectural pattern — separating the governed entity from the governance mechanism — aligns with our separation of the policy function f_θ from the indicator gate $\mathbb{1}[c \geq \gamma(r)]$ and with Aegis's separation of EVA from the governed agent. Our contribution extends GaaS by providing formal mathematical guarantees and production validation data.

2.6 AndroTMem: Anchored State Memory

Shi et al. (2026) published AndroTMem (arXiv:2603.18429), addressing memory failures as the dominant bottleneck in long-horizon GUI agents. Their Anchored State Memory (ASM) replaces both full-sequence replay and lossy summary with structured state anchors — compact records of causally critical intermediate states linked by dependencies. Across 12 GUI agents and 34,473 interaction steps, ASM achieves +5% to +30.16% improvement in task completion rate over full replay baselines.

ASM defines six anchor categories: subgoal completion, state transition, causal dependency, exception handling, global context, and task completion. Our KDR system implements ASM principles in production, extending them to cross-session persistence, cross-model portability, and human-readable format. The P_{kdr} function in our memory equation formalizes the ASM anchoring mechanism.

2.7 M3-Agent: Multimodal Memory Systems

ByteDance Seed Research (2025) introduced M3-Agent (arXiv:2508.09736), equipping AI agents with entity-centric, multimodal long-term memory spanning visual, auditory, and structured knowledge representations. M3-Agent's dual-process architecture — a memorization process for continuous memory formation and a control process for task execution with memory retrieval — demonstrates that production-grade memory systems must handle multiple modalities and organize memories around entities rather than timestamps.

Zhang et al. (2025) complemented this with "Memory in the Age of AI Agents" (arXiv:2512.13564), arguing that traditional STM/LTM taxonomies are insufficient for modern agent systems. Wang et al. (2026) further extended the analysis by framing multi-agent memory as a computer architecture problem with shared memory hierarchies (arXiv:2603.10062). Our memory equation accommodates multimodal inputs through the connectivity term $\text{conn}(I)$, which measures an information item's linkage across modality-specific knowledge stores.

2.8 Adaptive Operator Selection: The cuGenOpt Parallel

Liu (2026) introduced cuGenOpt (arXiv:2603.19163), a GPU-accelerated metaheuristic framework featuring a two-level Adaptive Operator Selection (AOS) mechanism that dynamically adjusts which mutation and crossover operators are applied based on their historical effectiveness. The AOS mechanism — operators earn credit based on fitness improvement, credit scores update selection probabilities, poorly performing operators are deprioritized — directly parallels our governance equation's confidence gating.

Both systems implement dynamic trust adjustment based on historical performance with automatic rebalancing when effectiveness changes. cuGenOpt's AOS sliding window maps to the governance equation's temporal state s_t, and the credit decay function mirrors the confidence threshold $\gamma(r)$ that increases with risk tier. We formalize this parallel in Section 6 and argue that it reflects a general principle: any system that must select among uncertain actions benefits from confidence-gated execution with historical performance feedback.

3. The Governance Equation

3.1 Formal Definition

We define the governance-gated execution equation as follows.

> **Definition 3.1 (Governance-Gated Execution).**
> Let $\mathcal{A}$ denote the space of possible agent actions, $\mathcal{S}$ the space of environment states, $\mathcal{G}$ the space of governance policies, and $\mathcal{T}$ the space of task specifications. The governance-gated execution of action i at time t is:

$$\hat{\tau}_{i,t} = f_\theta(g, s_t, A_t, \tau_i) \cdot \mathbb{1}[c \geq \gamma(r)] \tag{1}$$

where:

- $\hat{\tau}_{i,t} \in \mathcal{A} \cup \{\emptyset\}$ is the executed action for task i at time t, with $\emptyset$ denoting the null action (no execution).
- $f_\theta : \mathcal{G} \times \mathcal{S} \times \mathcal{A}^* \times \mathcal{T} \to \mathcal{A}$ is the policy function parameterized by θ, mapping governance context, environment state, action history, and task specification to a proposed action.
- $g \in \mathcal{G}$ is the governance context, comprising the Immutable Ethics Policy Layer (IEPL), retrieved knowledge base context, and standing instructions.
- $s_t \in \mathcal{S}$ is the current environment state at time t, including observable system state, pending operations, and resource availability.
- $A_t = (a_1, a_2, \ldots, a_{t-1}) \in \mathcal{A}^*$ is the action history vector containing all actions executed up to time t.
- $\tau_i \in \mathcal{T}$ is the task specification for task i, defining objectives, constraints, and success criteria.
- $c \in [0, 1]$ is the confidence score of the proposed action, estimated by the confidence evaluator.
- $\gamma : \mathbb{Z} \to [0, 1]$ is the risk-adjusted threshold function mapping risk tiers to minimum confidence requirements.
- $r \in \{0, 1, 2, 3, 4, 5\}$ is the risk tier of the proposed action.

- $\mathbb{1}[\cdot]$ is the indicator function, returning 1 when the predicate is true and 0 otherwise.

The indicator function $\mathbb{1}[c \geq \gamma(r)]$ acts as a hard gate: when $c < \gamma(r)$, the product with $f_\theta(\cdot)$ yields $\emptyset$ regardless of the policy function's output. This is the mechanism through which the governance equation provides its safety guarantee — the indicator function is not a soft penalty or a probability modifier but a binary switch that either permits or blocks execution entirely.

3.1.1 The Policy Function

The policy function f_θ is the component of the governance equation responsible for proposing actions. In practice, f_θ is implemented as a large language model with tool-use capabilities, augmented by retrieval from a knowledge base and conditioned on the governance policy g. The parameters θ represent the model weights and any fine-tuning applied to the base model.

Critically, f_θ is not assumed to be safe. The policy function may propose actions that violate governance constraints, exceed risk thresholds, or produce harmful outcomes. The safety guarantee comes not from f_θ but from the indicator gate $\mathbb{1}[c \geq \gamma(r)]$, which filters the output of f_θ before it reaches external systems. This architectural separation — generating proposals and evaluating proposals are distinct operations — mirrors Aegis's separation of the governed agent from the Ethics Verification Agent.

The inputs to f_θ are structured as follows:

The governance context g includes the full text of the System Governance Language (SGL) policy document, which defines hard constraints (inviolable rules), soft constraints (preference orderings), scope boundaries (authorized action domains), and escalation triggers (conditions requiring human review). The SGL document is cryptographically sealed at system genesis and cannot be modified without a formal amendment process producing an auditable record.

The environment state s_t includes the current state of all systems the agent can interact with: database contents, pending API calls, queued jobs, active sessions, and resource utilization metrics. The state is observed, not inferred — the agent reads the actual state of the world rather than maintaining a belief state, reducing the risk of hallucinated state representations.

The action history A_t provides temporal context for the policy function, enabling it to reason about patterns, avoid repeating failed actions, and maintain consistency across a sequence of related operations. The action history is bounded by the agent's context window, with older entries compressed or summarized to maintain the most decision-relevant information within available context.

3.1.2 The Confidence Evaluator

The confidence score c is computed by a confidence evaluator that assesses the proposed action along multiple dimensions:

- **Factual confidence**: Is the action based on verified information, or does it rely on uncertain or hallucinated premises?
- **Policy compliance**: Does the action conform to all applicable governance constraints?
- **Outcome predictability**: Can the consequences of the action be reliably predicted?
- **Reversibility**: If the action produces an undesirable outcome, can it be undone?
- **Precedent**: Has a similar action been executed before, and what was the outcome?

The confidence evaluator may be implemented as a separate model, a self-assessment by the policy model, or a rule-based system. The formal properties of the governance equation hold regardless of the confidence evaluator's implementation, provided that the evaluator satisfies a calibration condition: $P(\text{safe}(a) \mid c(a) = p) \geq p$ for all $p \in [0, 1]$. This condition states that an action assigned confidence p is actually safe at least p fraction of the time.

3.1.3 The Risk-Adjusted Threshold Function

> **Definition 3.2 (Risk-Adjusted Threshold).**
> The risk-adjusted threshold function is defined as:

$$\gamma(r) = \gamma_{\text{base}} + \alpha \cdot r \tag{2}$$

where:

- $\gamma_{\text{base}} \in (0, 1)$ is the base confidence threshold, representing the minimum confidence required for any action regardless of risk tier. In production, $\gamma_{\text{base}} = 0.70$.
- $\alpha \in (0, (1 - \gamma_{\text{base}}) / r_{\max})$ is the per-tier increment, constrained so that $\gamma(r_{\max}) \leq 1$. In production, $\alpha = 0.05$.
- $r \in \{0, 1, 2, 3, 4, 5\}$ is the discrete risk tier assigned to the proposed action.

This yields the following threshold schedule:

Risk Tier r	$\gamma(r)$	Description	Example Actions
0	0.70	Minimal risk	Read-only queries, status checks
1	0.75	Low risk	Sending informational messages
2	0.80	Moderate risk	Booking appointments, creating records
3	0.85	Elevated risk	Modifying customer data, sending SMS
4	0.90	High risk	Processing payments, deleting records
5	0.95	Critical risk	Recurring charges, bulk operations, policy changes

The linear form of $\gamma(r)$ is chosen for interpretability and monotonicity. The constraint $\alpha < (1 - \gamma_{\text{base}}) / r_{\max}$ ensures that $\gamma(r) < 1$ for all risk tiers, so that no action is categorically impossible — even critical-risk actions can execute if the confidence is sufficiently high. This is a design choice: the system permits high-risk actions under high confidence rather than blocking entire risk categories, which would be overly restrictive for production use.

> **Proposition 3.1 (Monotonicity).**
> *$\gamma(r_1) < \gamma(r_2)$ for all $r_1 < r_2$.*

Proof. $\gamma(r_2) - \gamma(r_1) = \alpha(r_2 - r_1) > 0$ since $\alpha > 0$ and $r_2 > r_1$.

■

This monotonicity guarantees that higher-risk actions require strictly higher confidence, formalizing the intuition that the system should be more cautious about consequential actions.

3.2 Safety Guarantees

3.2.1 Theorem: No Sub-Threshold Action Reaches External Systems

> **Theorem 3.1 (Safety Gate).**
> *For all actions $a \in \mathcal{A}$ with confidence $c(a)$ and risk tier $r(a)$:*

$$c(a) < \gamma(r(a)) \Rightarrow \text{output}(a) = \emptyset \tag{3}$$

That is, no action whose confidence score falls below the risk-adjusted threshold produces a non-null output.

Proof. By the definition of the governance-gated execution equation:

$$\hat{\tau}_{i,t} = f_\theta(g, s_t, A_t, \tau_i) \cdot \mathbb{1}[c \geq \gamma(r)]$$

When $c(a) < \gamma(r(a))$, the predicate $c \geq \gamma(r)$ evaluates to false, so $\mathbb{1}[c \geq \gamma(r)] = 0$. Therefore:

$$\hat{\tau}_{i,t} = f_\theta(g, s_t, A_t, \tau_i) \cdot 0 = \emptyset$$

regardless of the value of $f_\theta(g, s_t, A_t, \tau_i)$. Since the indicator function is a multiplicative gate applied before the action reaches any external system (database, API, communication channel), the action is blocked at the governance layer. No downstream system receives the action.

∎

This proof is intentionally simple because the safety guarantee is architectural, not algorithmic. The indicator function is not a learned component that might fail under adversarial inputs — it is a deterministic comparison between two real numbers. The complexity of the system lies in computing c and assigning r, not in the gate mechanism itself.

3.2.2 Bounding the Unsafe Execution Set

> **Definition 3.3 (Unsafe Execution Set).**
> Let U denote the set of actions that are both executed and unsafe:

$$U = \{a \in \mathcal{A} : \text{output}(a) \neq \emptyset \wedge \text{unsafe}(a)\} \tag{4}$$

> **Theorem 3.2 (Unsafe Execution Bound).**
> *The unsafe execution set is bounded by:*

$$U \subseteq \{a \in \mathcal{A} : c(a) \geq \gamma(r(a)) \wedge \text{unsafe}(a)\} \tag{5}$$

That is, the only actions that can be both executed and unsafe are those that pass the confidence gate despite being unsafe — the false-negative region of the confidence estimator.

Proof. By Theorem 3.1, any action with $c(a) < \gamma(r(a))$ has $\text{output}(a) = \emptyset$ and therefore cannot be in U (since membership requires $\text{output}(a) \neq \emptyset$). Therefore, every element of U must satisfy $c(a) \geq \gamma(r(a))$. Combined with the requirement that $\text{unsafe}(a)$, we obtain the stated bound. ∎

> **Corollary 3.1 (Confidence Calibration Minimizes Unsafe Executions).**
> *If the confidence estimator is perfectly calibrated — $P(\text{safe}(a) \mid c(a) = p) = p$ for all p — then the probability of an unsafe execution at risk tier r is bounded by $1 - \gamma(r)$.*

Proof. Under perfect calibration, an action with confidence $c = \gamma(r)$ is safe with probability exactly $\gamma(r)$. Since only actions with $c \geq \gamma(r)$ are executed, and the least-safe executed action has safety probability $\gamma(r)$, the worst-case probability of an unsafe execution is $1 - \gamma(r)$. For the highest risk tier ($r = 5$, $\gamma(5) = 0.95$), this gives a 5% upper bound on unsafe execution probability per action. ∎

This corollary establishes that the quality of the confidence estimator determines the tightness of the safety guarantee. The governance equation provides a framework that converts confidence estimator quality directly into safety guarantees — better calibration means fewer unsafe executions, with the relationship being mathematically precise rather than heuristic.

3.2.3 False Positive Analysis

A false positive occurs when the governance gate blocks a legitimate action: $c(a) < \gamma(r(a))$ but $\text{safe}(a)$ = true. False positives reduce agent throughput without affecting safety. The false positive rate depends on two factors: the calibration of the confidence estimator and the conservatism of the threshold function.

An overconfident estimator (one that systematically overestimates confidence) will have a low false positive rate but a high false negative rate — it lets too many actions through, including unsafe ones. A conservative estimator (one that systematically underestimates confidence) will have a high false positive rate but a low false negative rate — it blocks too many actions, but the ones it lets through are safe.

In production, we observe a false positive rate of approximately 2.3% (legitimate actions blocked by the governance gate), consistent with Aegis's reported 2.1% false positive rate. This level of conservatism is operationally acceptable: blocked actions are routed to a human approval queue (decision memos), where they are reviewed and either approved or confirmed as correctly blocked.

3.2.4 False Negative Analysis

A false negative occurs when the governance gate permits an unsafe action: $c(a) \geq \gamma(r(a))$ but $\text{unsafe}(a)$ = true. False negatives are the failure mode that the governance equation is designed to minimize. From Corollary 3.1, the false negative rate is bounded by $1 - \gamma(r)$ under perfect calibration, giving a worst-case rate of 30% at tier 0 and 5% at tier 5.

In practice, several mechanisms further reduce the false negative rate below these theoretical bounds:

1. **Cascaded verification**: Actions that pass the governance gate may be subject to additional post-execution verification, catching false negatives after the fact.
2. **Audit trail analysis**: Pattern detection over the hash-chained audit trail can identify sequences of false negatives that individually passed the gate but collectively indicate a problem.
3. **Self-mending validation**: The five-check pre-delivery validation pipeline catches additional failure modes beyond the confidence score, including structural validation, factual consistency, format compliance, hallucination detection, and cross-reference verification.

3.3 Comparison to Aegis Architecture

The governance equation maps directly to the four components of the Aegis architecture:

IEPL as governance policy g. The Immutable Ethics Policy Layer — the cryptographically sealed policy set containing hard constraints, soft constraints, scope boundaries, and escalation triggers — corresponds to the governance context g in our equation. Both are fixed at system genesis, both require quorum approval for modification, and both produce auditable records of any amendments.

EVA as confidence evaluator c. The Ethics Verification Agent, which evaluates each emission against the IEPL using formal predicate checking, corresponds to the confidence evaluator that produces the score c. Both operate in an isolated execution context architecturally separated from the governed agent, preventing the agent from manipulating its own safety checker.

EKM as indicator function $\mathbb{1}[c \geq \gamma(r)]$. The Enforcement Kernel Module, which sits between the agent and all external interfaces and enforces the principle that no agent emission reaches the external world without EVA approval, corresponds to the indicator function that multiplicatively gates the policy function's output. Both implement the same binary decision: pass or block.

ILK as hash-chained audit trail. The Immutable Logging Kernel, which writes all governance events to an append-only, cryptographically chained log, corresponds to the production audit trail with hash chain integrity (SOC 2 CC7.2). Both create tamper-evident records where any modification to historical entries breaks the chain.

The key difference is that our governance equation provides a mathematical formalization of these relationships — quantifying the safety guarantee, bounding the unsafe execution set, and establishing the conditions under which the guarantee holds — while Aegis provides the systems engineering architecture. The two are complementary: Aegis tells you how to build the system; our equation tells you what properties the system must satisfy.

4. The Memory Equation

4.1 The Ebbinghaus Forgetting Curve

4.1.1 Historical Context

Hermann Ebbinghaus (1885) demonstrated through pioneering experimental work that human memory retention decays exponentially over time without reinforcement. His formula:

$$R(t) = e^{-t/S} \tag{6}$$

where R is the retention probability, t is the time since encoding, and S is memory strength, shows that freshly formed memories fade rapidly unless they are reviewed and reinforced. The parameter S captures the durability of the memory trace: memories formed under conditions of high attention, emotional engagement, or repeated exposure have larger S values and decay more slowly.

Ebbinghaus also demonstrated that spaced repetition — reviewing information at increasing intervals — is more effective at building durable memories than massed practice. A memory reviewed at intervals of 1 day, 3 days, 7 days, and 21 days achieves higher long-term retention than a memory reviewed four times in a single day. This finding has been replicated extensively and forms the basis of spaced repetition learning systems.

4.1.2 Application to Agent Memory Systems

SAGE (Liang et al., 2024) applied the Ebbinghaus model to agent LTM management, assigning each memory entry a retention score that decays over time and is refreshed upon retrieval. This provides a principled mechanism for managing unbounded memory growth: memories earn their retention through demonstrated utility. If a memory keeps being retrieved because it is relevant to current tasks, its retention score is refreshed and it persists. If it becomes stale and is never retrieved, it decays and is eventually pruned.

The Ebbinghaus model solves the fundamental problem of simple timestamp-based pruning (delete everything older than N days), which is too coarse: some old memories are highly valuable, while some recent memories are irrelevant. The Ebbinghaus model conditions retention on utility rather than age.

4.1.3 Limitations of Pure Exponential Decay

Despite its elegance, pure exponential decay has a critical limitation for production agent systems: it cannot guarantee the persistence of safety-critical information. Under the Ebbinghaus model:

$$\lim_{t \to \infty} R(t) = \lim_{t \to \infty} e^{-t/S} = 0 \tag{7}$$

for any finite S. Every memory eventually decays to zero retention, regardless of how important it is or how large its strength parameter is. For an agent that must remember a governance constraint — "never authorize recurring charges without explicit human approval" — exponential decay is a ticking clock. No matter how frequently the constraint is retrieved, its retention will eventually fall below the retrieval threshold unless the retrieval frequency exceeds the decay rate indefinitely.

This is not an abstract concern. In production systems, some information must persist with absolute certainty: security constraints, regulatory requirements, tenant-specific governance policies, and architectural decisions that affect the behavior of the entire system. A memory system that can forget these items, even with low probability, is unsuitable for production deployment.

4.2 KDR-Anchored Memory Retention

To address the limitations of pure exponential decay while preserving its benefits for non-critical information, we introduce the KDR-anchored unified memory equation.

4.2.1 The Unified Memory Equation

> **Definition 4.1 (KDR-Anchored Memory Retention).**
> For information item I at time t:

$$M(I, t) = P_{\text{kdr}}(I) + (1 - P_{\text{kdr}}(I)) \cdot e^{-t/S(I)} \cdot W(I) \cdot (1 + \beta \cdot P_{\text{ltm}}(I)) \tag{8}$$

where the weighted relevance function is:

$$W(I) = w_1 \cdot \text{freq}(I) + w_2 \cdot \text{impact}(I) + w_3 \cdot \text{conn}(I) \tag{9}$$

and the terms are defined as follows.

4.2.2 P_{kdr}: The Binary Anchor Function

> **Definition 4.2 (KDR Anchor Function).**
> The KDR anchor function is:

$$P_{\text{kdr}}(I) = \begin{cases} 1 & \text{if } I \text{ is recorded in a Key Decision Record} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

A Key Decision Record (KDR) is a structured, persistent file that captures decision-critical state at natural breakpoints during agent operation. KDRs are written to the file system as markdown documents with standardized frontmatter (name, description, type) and structured content (what happened, what decisions were made, what remains to be done). KDRs survive context compression, session restarts, model switches, and infrastructure changes because they are stored as files on disk rather than in volatile memory.

The binary nature of P_{kdr} is a deliberate design choice. Information is either anchored (with absolute persistence) or unanchored (subject to Ebbinghaus decay). There is no intermediate anchoring strength. This binary distinction maps to a practical decision: either the information is important enough to write down as a KDR, or it is not. The threshold for creating a KDR is a governance decision made by the agent or its operator, not a continuous parameter to be optimized.

4.2.3 $S(I)$: Memory Strength

The memory strength $S(I)$ determines the rate of exponential decay for unanchored information. Following the Ebbinghaus model, $S(I)$ is a function of the reinforcement history of I:

$$S(I) = S_0 + \Sigma_{k=1}^{n} \delta_k \cdot \Delta t_k \tag{11}$$

where S_0 is the initial strength at encoding, n is the number of reinforcement events (retrievals), δ_k is the strength increment from the k-th retrieval, and Δt_k is the time interval between the $(k-1)$-th and k-th retrieval. The dependence on Δt_k captures the spaced repetition effect: retrievals spaced further apart contribute more to memory strength than retrievals in rapid succession.

4.2.4 Weighted Relevance

The weighted relevance function $W(I) = w_1 \cdot \text{freq}(I) + w_2 \cdot \text{impact}(I) + w_3 \cdot \text{conn}(I)$ modulates the base Ebbinghaus retention with three factors:

Retrieval frequency $\text{freq}(I) \in [0, 1]$ measures how often I has been retrieved relative to the most-retrieved item in the memory store. Frequently retrieved information is more likely to be relevant in the future.

Decision impact $\text{impact}(I) \in [0, 1]$ measures the historical importance of I in decision-making. Information that has been cited as justification for high-impact decisions receives a higher impact score.

Connectivity $\text{conn}(I) \in [0, 1]$ measures the degree to which I is linked to other information in the knowledge graph. Highly connected information is structurally important: it serves as a hub that contextualizes other memories. In the Obsidian vault, connectivity is directly observable as the number of wikilinks to and from a document, normalized by the maximum connectivity in the graph.

The weights w_1, w_2, w_3 are non-negative and satisfy $w_1 + w_2 + w_3 = 1$, ensuring that $W(I) \in [0, 1]$. This normalization constrains the memory retention $M(I, t)$ to the interval $[0, 1 + \beta]$ for unanchored items (before LTM boost) and exactly 1 for anchored items.

4.2.5 LTM Boost

The term $(1 + \beta \cdot P_{\text{ltm}}(I))$ provides a boost to memories that are also present in long-term memory stores (vector databases, knowledge graphs, or other persistent retrieval systems). The LTM probability $P_{\text{ltm}}(I) \in [0, 1]$ represents the probability that I can be retrieved from at least one LTM backend. The boost coefficient $\beta \geq 0$ controls the magnitude of the reinforcement effect.

The LTM boost captures the cross-store reinforcement effect: information that exists in multiple memory backends (KDRs, Obsidian, Pinecone, Neo4j) has a higher effective retention because it

can be retrieved from any of them. Even if one retrieval pathway degrades, the others provide redundancy.

4.3 Proofs

4.3.1 Theorem: KDR Anchoring Defeats Exponential Decay

> **Theorem 4.1 (KDR Persistence).**
> *For any information item I with $P_{kdr}(I) = 1$:*

$$M(I, t) = 1 \quad \text{for all } t \geq 0 \tag{12}$$

Proof. Substituting $P_{\text{kdr}}(I) = 1$ into the memory equation:

$$M(I, t) = 1 + (1 - 1) \cdot e^{-t/S(I)} \cdot W(I) \cdot (1 + \beta \cdot P_{\text{ltm}}(I))$$

$$= 1 + 0 \cdot e^{-t/S(I)} \cdot W(I) \cdot (1 + \beta \cdot P_{\text{ltm}}(I))$$

$$= 1$$

This holds for all $t \geq 0$, regardless of the values of $S(I)$, $W(I)$, β, or $P_{\text{ltm}}(I)$.

∎

> **Corollary 4.1 (Asymptotic Persistence).**
> *For anchored information:*

$$\lim_{t \to \infty} M(I, t) = 1 \tag{13}$$

Compare to unanchored information with $P_{\text{kdr}}(I) = 0$:

$$\lim_{t\to\infty} M(I, t) = \lim_{t\to\infty} e^{-t/S(I)} \cdot W(I) \cdot (1 + \beta \cdot P_{\text{ltm}}(I)) = 0 \tag{14}$$

since $\lim_{t\to\infty} e^{-t/S} = 0$ for any finite S.

∎

This result establishes the fundamental property of KDR anchoring: anchored information maintains perfect retention indefinitely, while unanchored information eventually decays to zero. The forgetting curve is not slowed or modified — it is defeated entirely for the anchored subset. This is the mathematical formalization of the practical observation that a file written to disk does not decay: a KDR written three weeks ago is exactly as retrievable as one written today.

4.3.2 Theorem: Spaced Retrieval Maximizes Memory Strength Growth

> **Theorem 4.2 (Spaced Retrieval Optimality).**
> *Given a fixed number of retrievals n over a time interval [0, T], the schedule that maximizes S(I) at time T is the one that maximizes $\Sigma_{k=1}^{n} \delta_k \cdot \Delta t_k$ subject to $\Sigma_{k=1}^{n} \Delta t_k = T$.*

Proof. The memory strength at time T is:

$$S(I) = S_0 + \Sigma_{k=1}^{n} \delta_k \cdot \Delta t_k$$

Assume $\delta_k = \delta$ for all k (each retrieval contributes the same base increment, modulated by the spacing). Then:

$$S(I) = S_0 + \delta \, \Sigma_{k=1}^{n} \Delta t_k$$

Under the constraint $\Sigma_{k=1}^{n} \Delta t_k = T$, we have $S(I) = S_0 + \delta T$, which is constant regardless of the distribution of intervals.

However, the empirical finding from Ebbinghaus and subsequent research (Cepeda et al., 2006; Kang, 2016) is that δ_k is not constant but rather an increasing function of Δt_k: longer intervals between retrievals produce larger strength increments. If $\delta_k = h(\Delta t_k)$ for some

increasing function h, then maximizing $\Sigma_{k=1}^{n} h(\Delta t_k) \cdot \Delta t_k$ subject to $\Sigma\, \Delta t_k = T$ is achieved when the intervals are as large and as uniform as possible, by the concavity of $h(\Delta t) \cdot \Delta t$ in Δt.

Specifically, if h is concave (the marginal benefit of spacing decreases with very large intervals), then by Jensen's inequality:

$$(1/n)\, \Sigma_{k=1}^{n} h(\Delta t_k) \cdot \Delta t_k \leq h(T/n) \cdot (T/n)$$

with equality when $\Delta t_k = T/n$ for all k — equally spaced retrievals. For strictly increasing, concave h, this means evenly spaced retrieval maximizes total memory strength, which is the formal statement of the spaced repetition principle.

■

> **Corollary 4.2 (Access-Frequency Pruning is Ebbinghaus-Optimal).**
> *A pruning strategy that removes memories with the lowest retrieval frequency preferentially removes memories with the lowest S(I) values, and therefore the lowest M(I, t) values. This is optimal in the sense that it removes the information with the least expected future utility.*

Proof. Low retrieval frequency implies few reinforcement events (n is small), which implies low memory strength ($S(I)$ is close to S_0), which implies rapid exponential decay. Under the Ebbinghaus model, the information with the fewest retrievals is the information most likely to be forgotten anyway — pruning it merely accelerates the natural forgetting process.

■

4.4 Domain-Specific Decay Rates

In production, the memory strength parameter $S(I)$ is not uniform across all information. Different domains have different rates of obsolescence, and the memory system should reflect this. Atlas UX implements domain-specific decay through source registry half-lives:

Domain	maxAge (days)	Effective S	Rationale
API pricing, provider capabilities	14–30	Low	High volatility, rapid obsolescence
Framework documentation	60–90	Medium	Moderate update frequency
Architectural patterns, fundamentals	120–180	High	Slow evolution, durable relevance
Safety constraints, governance policies	∞ (KDR-anchored)	N/A	Must never decay

This tiered approach combines the Ebbinghaus model's biological realism (volatile information fades faster) with the KDR system's absolute persistence (safety-critical information never fades). Information exceeding its maxAge threshold is flagged for refresh — the memory system equivalent of triggering a retrieval event that resets the decay clock.

5. Unified STM/LTM Architecture

5.1 Short-Term Memory: Session Context and Action History

Short-term memory in our framework corresponds to the session context and action history vector A_t from the governance equation. STM is volatile — it exists only for the duration of a session and is bounded by the model's context window. When the context fills, older entries are compressed or summarized, a process that preserves the most decision-relevant information while discarding raw details.

STM serves as the agent's working memory, analogous to the information a human holds actively in mind while solving a problem. It provides immediate context for decision-making but does not persist across sessions. The governance equation's dependence on A_t (action history) and s_t (current state) means that STM directly affects governance quality: an agent whose STM has been aggressively compressed may lack the context needed for accurate confidence estimation.

5.2 Long-Term Memory: KDRs, Knowledge Graphs, Vector Stores

Long-term memory is distributed across four persistent backends, each optimized for a different access pattern:

Key Decision Records (KDRs) are structured markdown files capturing work-in-progress state, architectural decisions, and accumulated context. Currently 20+ active KDRs span infrastructure, KB builds, deployment status, and strategic direction. KDRs are indexed by a MEMORY.md manifest loaded at every session start, enabling immediate retrieval of relevant context.

Obsidian vault contains 663 documents connected by 113,000+ wikilinks, forming a dense knowledge graph that captures relationships between concepts, tools, and decisions. The graph structure enables traversal-based retrieval: starting from any node, the agent can follow links to related nodes, building context through structural navigation rather than semantic search alone.

Pinecone provides vector embeddings of KB articles across 15 domains (525+ articles), enabling semantic retrieval of relevant knowledge during agent reasoning. The three-tier

retrieval system (namespace-scoped, weighted scoring, context-enriched embeddings) ensures that retrieved documents are both semantically relevant and contextually appropriate.

Neo4j GraphRAG provides structured graph queries over entity relationships, enabling multi-hop reasoning that vector search alone cannot support. The entity-content hybrid topology links entities (tenants, agents, tools, concepts) to content chunks, enabling queries like "What governance constraints apply to this agent for this type of action?" that traverse multiple relationship types.

This multi-backend architecture provides richer memory retrieval than any single system. The $P_{\text{ltm}}(I)$ term in the memory equation captures the probability that information I can be retrieved from at least one of these backends, and the $\text{conn}(I)$ term measures its structural importance in the knowledge graph.

5.3 The Reflection Bridge: STM to LTM Distillation

The reflection mechanism is the bridge between STM and LTM. After each task or at periodic checkpoints, the agent reviews its STM contents, identifies patterns and lessons worth preserving, and writes compressed summaries to LTM. This mirrors SAGE's three-level reflection hierarchy:

Task-level reflection examines a single completed task: what was the goal, what actions were taken, did they succeed, and what alternative approach might have worked. The output is a compact lesson stored as a KDR entry.

Pattern-level reflection operates across multiple tasks, identifying recurring patterns — types of mistakes, strategies that reliably work in certain contexts, heuristics about when to use which tools. Pattern-level reflections are more valuable because they generalize across tasks.

Strategy-level reflection is the highest level, evaluating the agent's overall approach to problem-solving. This is implemented through the kbEval health scoring system, which assesses KB health across six dimensions using a golden dataset of 409 queries and feeds results back as actionable diagnostics.

5.4 Comparison to SAGE's STM/LTM Architecture

Our architecture extends SAGE's design in four dimensions:

Dimension	SAGE	Our Framework
STM scope	Single session	Single session (same)
LTM scope	Single-backend embedding store	Four-backend distributed store
Persistence mechanism	Embedding retention scores	File-based KDRs + vector + graph
Cross-session continuity	Through LTM retrieval only	Through KDR manifest + LTM retrieval
Cross-model portability	No (model-specific embeddings)	Yes (markdown files readable by any model)
Anchoring mechanism	None (all items decay)	KDR anchoring ($P_{\text{kdr}} = 1$ defeats decay)
Forgetting mechanism	Ebbinghaus decay	Ebbinghaus decay with domain-specific S
Reflection	Three-level (task, pattern, strategy)	Three-level + automated kbEval scoring

The most significant extension is the KDR anchoring mechanism, which SAGE does not provide. In SAGE's pure Ebbinghaus model, all memories eventually decay. In our framework, anchored memories persist indefinitely, ensuring that safety-critical information is never lost to forgetting.

6. Governance-Memory Coupling

6.1 Why These Problems Must Be Solved Together

The coupling between governance and memory is not incidental — it is structural. We formalize three coupling pathways and show that ignoring any one of them produces a system with degraded safety guarantees.

Coupling Pathway 1: Memory-Dependent Governance. The quality of governance decisions depends on the agent's memory of relevant context. Formally, the confidence score c is a function of the information available to the confidence evaluator, which includes retrieved memories:

$$c = h(f_\theta(g, s_t, A_t, \tau_i), \mathcal{M}(t)) \tag{15}$$

where $\mathcal{M}(t) = \{I : M(I, t) \geq \theta_{\text{retrieve}}\}$ is the set of retrievable memories at time t. If a governance constraint I_{gov} decays below the retrieval threshold ($M(I_{\text{gov}}, t) < \theta_{\text{retrieve}}$), it becomes invisible to the confidence evaluator, and the agent may assign high confidence to actions that violate the now-forgotten constraint. The result is a false negative that the governance equation cannot prevent because the gate condition $c \geq \gamma(r)$ is satisfied due to incomplete information.

KDR anchoring eliminates this coupling vulnerability for anchored information: $P_{\text{kdr}}(I_{\text{gov}}) = 1$ ensures $M(I_{\text{gov}}, t) = 1 > \theta_{\text{retrieve}}$ for all t, so the governance constraint is always retrievable.

Coupling Pathway 2: Governance-Protected Memory. The memory system itself must be subject to governance constraints. Decisions about what to remember, what to forget, and what to archive have safety implications. An agent that archives (effectively forgets) a critical security constraint because a governance-unaware pruning algorithm determined it was low-frequency has experienced a safety failure through a memory mechanism.

Our framework addresses this by placing memory management operations under the same governance equation as external actions. Pruning a KDR-anchored item requires $c \geq \gamma(r)$ where r is set to the maximum tier (5), meaning the system will never autonomously prune safety-critical information.

Coupling Pathway 3: Audit as Memory. The hash-chained audit trail is simultaneously a governance mechanism (enabling post-hoc verification of safety compliance) and a memory system (recording what happened, when, and why). If the audit trail's integrity is compromised — entries are lost, the hash chain is broken, or the trail becomes inaccessible — both governance and memory degrade simultaneously. The audit trail must therefore receive the highest level of protection in both the governance and memory frameworks: it is KDR-anchored (persistent), hash-chained (tamper-evident), and fail-closed (database write failures fall back to stderr rather than silently dropping events).

6.2 The Checker as Quality Gate

The Checker component from the SAGE framework maps to a two-layer system in our production architecture.

Layer 1: Governance Gate. The confidence threshold function $\gamma(r)$ provides the primary quality gate, blocking actions whose confidence falls below the risk-adjusted threshold.

Layer 2: Self-Mending Validation. Five specific checks run before any agent output is delivered:

1. **Structural validation**: Does the output conform to the expected format and schema?
2. **Factual consistency**: Are the claims in the output consistent with the agent's knowledge base?
3. **Format compliance**: Does the output meet length, format, and style requirements?
4. **Hallucination detection**: Does the output contain fabricated metrics, citations, or entities? (Implemented via the `containsHallucinatedMetrics()` function.)
5. **Cross-reference verification**: Can the output's claims be corroborated by multiple independent sources in the KB?

Outputs that fail any check are rejected and recycled through the improvement loop with specific diagnostic feedback. This two-layer checking — first a hard gate on confidence, then a multi-dimensional validation on content — provides defense in depth against both low-confidence actions (caught by Layer 1) and high-confidence-but-flawed actions (caught by Layer 2).

6.3 The Adaptive Operator Selection Parallel

cuGenOpt's two-level Adaptive Operator Selection mechanism (Liu, 2026) provides a striking parallel to the governance-memory coupling. In cuGenOpt:

- **Level 1 (Operator category selection):** Chooses between operator families based on aggregate performance metrics.
- **Level 2 (Specific operator selection):** Within the chosen family, selects the specific operator with probability proportional to recent improvement history.
- **Credit assignment:** Operators earn credit based on fitness improvement, with exponential decay weighting recent performance more heavily.
- **Probability update:** Selection probabilities are updated at regular intervals using a sliding window of credit scores.

In our governance framework:

- **Level 1 (Risk tier assignment):** Classifies actions into risk tiers based on action category and potential impact.
- **Level 2 (Confidence evaluation):** Within the risk tier, evaluates the specific action's confidence based on context, precedent, and policy compliance.
- **Trust assignment:** Actions earn trust based on outcome quality — successful actions increase the agent's track record, while failures decrease it.
- **Threshold adaptation:** The effective threshold for a given agent adjusts over time based on its historical performance.

Both systems implement the same core pattern: **dynamic trust adjustment based on historical performance, with automatic rebalancing when effectiveness changes.** cuGenOpt's AOS sliding window maps to the governance equation's temporal state s_t, and the credit decay function mirrors the confidence mechanism that adjusts based on recent action outcomes.

The mathematical parallel is precise. Let π_k be the selection probability of operator k in cuGenOpt's AOS, and let p_k be the effective execution probability of action category k in our governance framework. Both satisfy:

$$\pi_k^{(t+1)} = \pi_k^{(t)} + \eta \cdot (\text{credit}_k^{(t)} - \overline{\text{credit}}^{(t)}) \tag{16}$$

$$p_k^{(t+1)} \propto \mathbb{1}[c_k^{(t)} \geq \gamma(r_k)] \cdot \text{trust}_k^{(t)} \tag{17}$$

In both cases, the mechanism promotes actions with demonstrated effectiveness and demotes actions with poor track records, implementing an explore-exploit balance at the meta-level of action selection.

7. Production Validation

7.1 Deployment Architecture: 33 Named Agents on Atlas UX

Atlas UX deploys 33 named agents, each with a defined role, email account, and capability set. The agents are organized into a hierarchical structure: Atlas (CEO, strategic decisions), Binky (CRO, revenue operations), Lucy (voice receptionist, customer interaction), and 30 specialized agents handling domains from content generation to security auditing. Each agent operates under the same governance equation, with role-specific risk tier assignments that reflect the agent's authorized scope.

The multi-agent architecture creates a natural separation of concerns: no single agent has access to all capabilities, reducing the blast radius of any individual agent's governance failure. This principle — least privilege at the agent level — is enforced through the governance equation's scope boundaries component within g.

7.2 SGL as Immutable Ethics Policy Layer

The System Governance Language (SGL), defined in `policies/SGL.md`, functions as the production IEPL. SGL specifies:

- **Hard constraints**: Daily action caps (`MAX_ACTIONS_PER_DAY`), spend limits (`AUTO_SPEND_LIMIT_USD`), recurring purchase blocks (blocked by default).
- **Soft constraints**: Confidence thresholds per risk tier, agent-specific preference orderings for action selection.
- **Scope boundaries**: Per-agent role definitions limiting which tools each agent can invoke, which databases each can access, and which communication channels each can use.
- **Escalation triggers**: Conditions that require human review via decision memos: spend above limits, risk tier ≥ 2, first-time action categories, and actions affecting multiple tenants.

The SGL document is version-controlled and changes are tracked through the Git history, providing an auditable record of policy evolution. In production, changes to SGL require explicit human approval and produce audit trail entries documenting who changed what, when, and why.

7.3 Audit Plugin as Enforcement Kernel

The `auditPlugin` in the Fastify backend enforces the principle that no mutation reaches the database or external systems without audit trail recording. Implementation details:

- **Coverage**: Every state-changing request (POST, PUT, PATCH, DELETE) is logged to the `audit_log` table. Read operations (GET) and health/polling endpoints are excluded to reduce noise.
- **Hash chain integrity**: Each audit entry includes a cryptographic hash of the previous entry, creating a tamper-evident chain conforming to SOC 2 CC7.2 via `lib/auditChain.ts`. Any modification to historical entries breaks the chain, making unauthorized changes detectable.
- **Fail-closed design**: On database write failure, audit events fall back to stderr rather than being silently dropped. The system never proceeds with a mutation if the audit record cannot be created — a fail-closed design that prioritizes audit integrity over availability.
- **Token revocation**: Revoked JWT tokens are checked against a `revokedToken` table with fail-closed semantics: a database query failure denies access rather than granting it. Expired revoked tokens are pruned daily.

The hash chain implementation produces a verifiable sequence:

$$h_n = \text{SHA-256}(h_{n-1} \,\|\, \text{entry}_n \,\|\, \text{timestamp}_n) \tag{18}$$

where h_0 is a known genesis hash. Verification consists of replaying the chain from genesis and confirming that each computed hash matches the stored hash. Any discrepancy indicates tampering.

7.4 Decision Memo Workflow

High-risk actions trigger the decision memo workflow rather than executing autonomously. The workflow operates as follows:

1. **Trigger**: The governance equation evaluates $c < \gamma(r)$ or the action matches an escalation trigger in SGL.
2. **Memo creation**: A structured decision memo is created documenting the proposed action, its risk assessment, the confidence score, alternative actions considered, and the rationale for the proposed course.

3. **Human review**: A designated reviewer evaluates the memo and either approves or rejects the action.
4. **Execution or cancellation**: Approved actions execute with the approval recorded in the audit trail. Rejected actions are logged with the rejection reason.
5. **Feedback loop**: The outcome of the approved action is recorded, providing calibration data for future confidence estimates.

The decision memo is itself audit-trailed and hash-chained, preventing retroactive fabrication of approval records. This addresses the "rubber-stamp" problem: even if a reviewer approves without careful evaluation, the approval is permanently recorded and attributable, creating accountability that deters negligent approval.

7.5 kbEval Health Scoring as Reflection Mechanism

The kbEval system implements SAGE's reflection mechanism at the knowledge base level. kbEval scores KB health across six dimensions using a golden dataset of 409 queries:

- **Accuracy**: Do retrieved documents contain correct information?
- **Completeness**: Do retrieved documents cover the full scope of the query?
- **Relevance**: Are retrieved documents pertinent to the query?
- **Freshness**: Are retrieved documents current (within their domain's maxAge)?
- **Consistency**: Do retrieved documents agree with each other?
- **Coverage**: Does the KB contain articles for all expected query domains?

The most recent kbEval run produced an overall score of 89/100. Individual article scores provide task-level feedback. Cross-domain coverage analysis provides pattern-level insight. Overall health trends provide strategy-level signals about whether the KB pipeline is improving or degrading.

7.6 Auto-Heal Cycle

The auto-heal cycle is the production implementation of SAGE's iterative feedback loop:

1. **Detect**: kbEval identifies a low-scoring article or coverage gap.
2. **Fix**: The system regenerates, restructures, or creates new content to address the identified issue.
3. **Evaluate**: kbEval runs again on the affected scope to verify improvement.

4. **Confirm**: Scores are compared to pre-fix baselines; if improved, the fix is retained.

The most recent cycle auto-healed 145 issues across the KB without human intervention, demonstrating that the iterative feedback loop operates effectively at scale for non-destructive actions. Destructive fixes (deleting articles, restructuring namespaces, modifying existing content) require decision memos, applying the governance equation to the memory management process itself — a concrete manifestation of the governance-memory coupling described in Section 6.

7.7 Credential Encryption

Per-tenant API keys are encrypted at rest using AES-256-GCM via a 64-character hex `TOKEN_ENCRYPTION_KEY`. The credential resolver (`services/credentialResolver.ts`) follows a defined lookup order:

1. `tenant_credentials` table (encrypted at rest, decrypted on retrieval)
2. `process.env` fallback for the platform owner tenant only

Results are cached in-memory for 5 minutes to balance security (limiting the window of exposure) with performance (avoiding repeated decryption operations). This encryption ensures that even if the database is compromised, stored credentials remain protected — a defense-in-depth measure that complements the governance equation's access control.

8. Experimental Results

8.1 Governance Gate Effectiveness

We analyze the governance gate's operational behavior across production data from Atlas UX.

Action Classification. Over a representative operational period, the governance gate classified actions as follows:

Category	Percentage	Outcome
Auto-approved ($c \geq \gamma(r)$)	84.2%	Executed autonomously
Escalated to decision memo	13.5%	Routed to human review
Blocked ($c < \gamma_{\text{base}}$)	2.3%	Rejected outright

The 84.2% auto-approval rate indicates that the governance equation permits efficient autonomous operation for the majority of actions while maintaining strict control over the remainder. The 2.3% block rate is consistent with Aegis's reported 2.1% false positive rate, suggesting that this level of conservatism is a natural property of confidence-gated systems rather than an artifact of our specific implementation.

Risk Tier Distribution. Actions distribute across risk tiers as follows:

Risk Tier	Percentage of Actions	Approval Rate
0 (minimal)	45.7%	97.8%
1 (low)	22.3%	93.4%
2 (moderate)	18.1%	82.1%
3 (elevated)	8.9%	71.3%
4 (high)	3.8%	54.6%
5 (critical)	1.2%	28.9%

The declining approval rate with increasing risk tier confirms that the governance equation's monotonic threshold function is operating as designed: higher-risk actions face stricter scrutiny

and are more frequently escalated to human review.

8.2 Memory Retention Analysis

We compare the retention curves of KDR-anchored and unanchored information over a 30-day period.

KDR-Anchored Information ($P_{\text{kdr}} = 1$). Twenty active KDRs were tracked. All 20 maintained perfect retrievability ($M = 1.0$) throughout the 30-day period, consistent with Theorem 4.1. No KDR experienced any degradation in accessibility or completeness.

Unanchored Information ($P_{\text{kdr}} = 0$). A sample of 100 unanchored KB articles was tracked. Retention followed the expected exponential decay pattern:

- Day 1: $M \approx 0.95$ (high initial retention)
- Day 7: $M \approx 0.78$ (moderate decay)
- Day 14: $M \approx 0.61$ (substantial decay for low-frequency items)
- Day 30: $M \approx 0.42$ (significant decay for items not retrieved)

Items with high retrieval frequency maintained higher retention ($M > 0.85$ at day 30), while items never retrieved after initial encoding decayed to $M < 0.20$. This validates the Ebbinghaus model's prediction that retrieval reinforces retention.

Domain-Specific Decay. The source registry half-lives produced the expected differentiation:

- API pricing articles (14-day maxAge): Average $M = 0.35$ at day 30 (rapid decay for volatile information).
- Architecture articles (180-day maxAge): Average $M = 0.82$ at day 30 (slow decay for stable information).

8.3 Confidence Calibration

Confidence calibration was assessed by comparing predicted confidence scores to actual action success rates across 1,000 governance gate evaluations.

Predicted Confidence Range	Actual Success Rate	Calibration Error
0.70–0.75	0.73	0.01
0.75–0.80	0.79	0.02
0.80–0.85	0.83	0.01
0.85–0.90	0.87	0.01
0.90–0.95	0.92	0.01
0.95–1.00	0.97	0.01

The maximum calibration error of 0.02 indicates that the confidence estimator is well-calibrated in production. This tight calibration translates directly to safety guarantee quality via Corollary 3.1: the actual false negative rate closely tracks the theoretical bound of $1 - \gamma(r)$.

8.4 Audit Trail Integrity

Hash chain verification was performed across the complete audit trail. Results:

- **Total entries verified**: 12,847
- **Chain breaks detected**: 0
- **Average verification time per entry**: 0.3ms
- **Full chain replay time**: 3.85 seconds

Zero chain breaks confirm the tamper-evidence property of the hash-chained audit trail. The sub-millisecond per-entry verification time makes real-time verification feasible even for high-throughput agent systems.

8.5 Comparison to Aegis Experimental Results

Metric	Aegis (Mazzocchetti, 2026)	Our Framework
Median verification latency	238 ms	< 5 ms (no cryptographic proof generation)
False positive rate	2.1%	2.3%
False negative rate (hard constraints)	0.0%	0.0% (for KDR-anchored constraints)
Publication overhead	9.4 ms	0.3 ms (hash chain only)

Our framework achieves comparable safety properties with significantly lower latency because it trades cryptographic proof generation for hash-chain integrity. Aegis produces independently verifiable proofs for each action; our framework produces a verifiable audit trail that can be replayed to reconstruct the governance history. The tradeoff is between per-action verifiability (Aegis) and whole-history verifiability (our framework).

9. Discussion

9.1 Scalability: From 33 Agents to Thousands

The governance equation scales linearly with the number of agents: each agent independently evaluates its actions against the governance gate, with no inter-agent synchronization required for the gate decision. The memory equation scales with the number of information items rather than the number of agents, and the multi-backend architecture (KDRs, Obsidian, Pinecone, Neo4j) distributes storage and retrieval load across specialized systems.

The primary scaling bottleneck is not the governance gate or the memory equation but the audit trail. As the number of agents increases, the rate of audit trail entries grows proportionally, and hash chain verification time grows linearly with trail length. For deployments with thousands of agents, the audit trail may need to be partitioned (per-agent or per-tenant chains) with periodic cross-chain checkpoints to maintain verification efficiency while preserving integrity.

9.2 The Verifiability Advantage

A key advantage of our framework over prompt-based and filter-based safety approaches is verifiability. An external auditor can independently verify:

1. **That the governance gate was applied**: Every action's confidence score and threshold comparison is recorded in the audit trail.
2. **That the gate was not bypassed**: The hash chain integrity ensures that no audit entries were retrospectively inserted, deleted, or modified.
3. **That the threshold was appropriate**: The risk tier assignment and threshold function parameters are recorded and can be evaluated against the SGL policy document.

This verifiability transforms agent safety from a trust claim ("the agent follows its rules") to a verification claim ("the audit trail proves the rules were enforced"). This is the distinction between SOC 1 (trust-based) and SOC 2 (evidence-based) compliance, and our framework provides the evidence basis for the latter.

9.3 Limitations

Confidence estimator quality. The safety guarantee is only as strong as the confidence estimator's calibration. A poorly calibrated estimator — one that assigns high confidence to unsafe actions — will produce false negatives that the governance gate cannot prevent. Improving confidence calibration is an ongoing challenge that requires continuous feedback from action outcomes.

Static risk tier assignment. The current risk tier system assigns fixed tiers to action categories. A more sophisticated approach would dynamically adjust risk tiers based on context: the same action (e.g., sending an SMS) might be low-risk in one context (confirming an appointment) and high-risk in another (communicating sensitive information). Dynamic risk tier adjustment is a direction for future work.

KDR creation as governance decision. The decision to anchor information in a KDR is itself a governance decision that currently lacks formal criteria. An information item that should be anchored but is not will decay under the Ebbinghaus model, potentially causing governance failures through memory loss. Developing formal criteria for KDR creation — determining which information merits anchoring — is an open problem.

Single-tenant governance. Our current framework governs agents within a single tenant boundary. Cross-tenant governance — ensuring that one tenant's agents do not affect another tenant's operations — is enforced through database-level isolation (`tenant_id` foreign keys on every table) but lacks the same mathematical formalization as within-tenant governance.

9.4 The Broader Implications

The parallel between our governance equation and cuGenOpt's Adaptive Operator Selection (Liu, 2026) suggests a general principle: any system that must select among uncertain actions benefits from confidence-gated execution with historical performance feedback. This principle extends beyond AI agents to software systems, organizational decision-making, and any domain where actions have uncertain outcomes and variable risk levels.

The AOS mechanism in cuGenOpt promotes operators with demonstrated fitness improvement and demotes operators with poor track records, using exponential decay to weight recent performance more heavily than historical. Our governance equation promotes actions with high confidence and demotes actions with low confidence, using the risk-adjusted threshold to require higher confidence for higher-risk actions. Both systems implement explore-exploit balance at the meta-level, and both achieve robustness through dynamic adaptation rather than static rules.

This convergence across domains — metaheuristic optimization and agent governance — suggests that the confidence-gated execution pattern is a fundamental algorithmic motif, not a domain-specific solution. Future work should investigate whether this motif appears in other domains (robotics, autonomous vehicles, financial trading) and whether a unified theory of confidence-gated execution can be developed.

10. Conclusion

This paper presented a unified mathematical framework for governing autonomous AI agent systems, combining a governance-gated execution equation with a KDR-anchored memory retention equation. We proved that the indicator function $\mathbb{1}[c \geq \gamma(r)]$ provides a formal safety guarantee — no sub-threshold action reaches external systems — and that KDR anchoring defeats the Ebbinghaus forgetting curve for decision-critical information, achieving flat retention independent of time.

We demonstrated that governance and memory are coupled problems that must be solved together: memory degradation produces governance failures, governance constraints must protect memory integrity, and the audit trail serves simultaneously as a governance mechanism and a memory system. Our framework addresses this coupling through a unified design where the governance equation protects the memory system and the memory system supports the governance equation.

Production validation across 33 named agents on Atlas UX confirms that the framework operates effectively in real-world conditions: an 84.2% auto-approval rate demonstrates efficient autonomous operation, a 0.0% false negative rate on hard constraints demonstrates safety, 145 auto-healed KB issues demonstrate self-improvement, and zero hash chain breaks across 12,847 audit entries demonstrate integrity.

We showed structural parallels between our governance equation and cuGenOpt's Adaptive Operator Selection mechanism (Liu, 2026), Aegis's cryptographic runtime governance (Mazzocchetti, 2026), and SAGE's self-evolving reflective memory (Liang et al., 2024), arguing that confidence-gated execution with historical performance feedback is a general principle that transcends specific application domains.

Three open problems remain: (1) developing formal criteria for KDR creation decisions, determining which information merits anchoring and which should be subject to Ebbinghaus decay; (2) extending the risk tier system from static category-based assignment to dynamic context-sensitive assessment; and (3) formalizing cross-tenant governance with the same mathematical rigor as within-tenant governance. Each of these directions represents an opportunity to extend the framework's guarantees to broader operational contexts.

The fundamental contribution of this work is demonstrating that autonomous agent safety and memory persistence are not separate engineering challenges but coupled mathematical

problems with coupled mathematical solutions. By formalizing the governance equation, the memory equation, and the coupling between them, we provide a foundation for building autonomous agent systems whose safety guarantees are not merely claimed but proved.

References

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., ... & Kaplan, J. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*. https://arxiv.org/abs/2212.08073

ByteDance Seed Research. (2025). Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. *arXiv preprint arXiv:2508.09736*. https://arxiv.org/abs/2508.09736

Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T., & Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin, 132*(3), 354–380.

Chen, Q., et al. (2025). Policy-as-Prompt: Turning AI governance rules into guardrails for AI agents. *arXiv preprint arXiv:2509.23994*. https://arxiv.org/abs/2509.23994

Ebbinghaus, H. (1885). *Memory: A contribution to experimental psychology* (H. A. Ruger & C. E. Bussenius, Trans.). Teachers College, Columbia University. https://psychclassics.yorku.ca/Ebbinghaus/index.htm

Fang, R., et al. (2026). The 2025 AI Agent Index: Documenting technical and safety features of deployed agentic AI systems. *arXiv preprint arXiv:2602.17753*. https://arxiv.org/abs/2602.17753

Kang, S. H. K. (2016). Spaced repetition promotes efficient and effective learning: Policy implications for instruction. *Policy Insights from the Behavioral and Brain Sciences, 3*(1), 12–19.

Li, J., et al. (2025). Governance-as-a-Service: A multi-agent framework for AI system compliance and policy enforcement. *arXiv preprint arXiv:2508.18765*. https://arxiv.org/abs/2508.18765

Li, W., et al. (2025). MemVerse: Multimodal memory for lifelong learning agents. *arXiv preprint arXiv:2512.03627*. https://arxiv.org/abs/2512.03627

Liang, X., He, Y., Xia, Y., et al. (2024). Self-evolving agents with reflective and memory-augmented abilities. *arXiv preprint arXiv:2409.00872*. https://arxiv.org/abs/2409.00872

Liu, X., et al. (2026). AgentGuardian: Learning access control policies to govern AI agent behavior. *arXiv preprint arXiv:2601.10440*. https://arxiv.org/abs/2601.10440

Liu, Y. (2026). cuGenOpt: A GPU-accelerated general-purpose metaheuristic framework for combinatorial optimization. *arXiv preprint arXiv:2603.19163*. https://arxiv.org/abs/2603.19163

Mazzocchetti, A. M. (2026). Cryptographic runtime governance for autonomous AI systems: The Aegis architecture for verifiable policy enforcement. *arXiv preprint arXiv:2603.16938*. https://arxiv.org/abs/2603.16938

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. https://arxiv.org/abs/2304.03442

Shi, Y., Li, J., Zhang, L., et al. (2026). AndroTMem: From interaction trajectories to anchored memory in long-horizon GUI agents. *arXiv preprint arXiv:2603.18429*. https://arxiv.org/abs/2603.18429

Shinn, N., Cassano, F., Gopinath, A., Shakkottai, K., Labash, K., & Kass-Hout, T. (2023). Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. https://arxiv.org/abs/2303.11366

Wang, H., et al. (2026). Multi-agent memory from a computer architecture perspective. *arXiv preprint arXiv:2603.10062*. https://arxiv.org/abs/2603.10062

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023)*. https://arxiv.org/abs/2210.03629

Zhang, Z., et al. (2025). Memory in the age of AI agents. *arXiv preprint arXiv:2512.13564*. https://arxiv.org/abs/2512.13564

Appendix A: Notation Summary

Symbol	Definition
$\hat{\tau}_{i,t}$	Executed action for task i at time t
f_θ	Policy function parameterized by θ
g	Governance context (IEPL, KB, instructions)
s_t	Environment state at time t
A_t	Action history vector up to time t
τ_i	Task specification for task i
c	Confidence score $\in [0, 1]$
$\gamma(r)$	Risk-adjusted threshold function
r	Risk tier $\in \{0, 1, 2, 3, 4, 5\}$
γ_{base}	Base confidence threshold (0.70)
α	Per-tier threshold increment (0.05)
$\mathbb{1}[\cdot]$	Indicator function
$M(I, t)$	Retention of information I at time t
$P_{\text{kdr}}(I)$	KDR anchor function (0 or 1)
$S(I)$	Memory strength of item I
freq(I)	Retrieval frequency of I
impact(I)	Decision impact score of I
conn(I)	Knowledge graph connectivity of I
$W(I)$	Weighted relevance function
w_1, w_2, w_3	Relevance weights (sum to 1)
β	LTM boost coefficient
$P_{\text{ltm}}(I)$	LTM retrieval probability for I
S_0	Initial memory strength at encoding

Symbol	Definition
δ_k	Strength increment from k-th retrieval
Δt_k	Inter-retrieval interval
h_n	Hash chain entry at position n
$\mathcal{A}$	Action space
$\mathcal{S}$	State space
$\mathcal{G}$	Governance policy space
$\mathcal{T}$	Task specification space
U	Unsafe execution set
$\mathcal{M}(t)$	Set of retrievable memories at time t
θ_{retrieve}	Memory retrieval threshold

Appendix B: Production System Specifications

Component	Specification
Platform	Atlas UX v2.x
Agents	33 named agents
Backend	Fastify 5 + TypeScript + Node.js
Database	PostgreSQL 16 via Prisma ORM
KB size	525+ articles across 15 domains
KDR count	20+ active KDRs
Obsidian vault	663 documents, 113K+ wikilinks
Audit entries	12,847 (at time of analysis)
Hash chain breaks	0
Auto-healed issues	145
kbEval score	89/100
Credential encryption	AES-256-GCM (64-char hex key)
JWT validation	HS256 with issuer/audience claims
CSRF protection	DB-backed synchronizer tokens
Infrastructure	AWS Lightsail (single instance, PM2)