Biological Motifs for Agentic Control

A Categorical Isomorphism between Gene Regulatory Networks and Autonomous Software Architectures

Preprint – Feedback Welcome

Bogdan Banu
bogdan@banu.be

February 14, 2026

Abstract

The transition of Large Language Models (LLMs) from passive generators to autonomous agents has introduced significant challenges in reliability, security, and state management. Current agentic architectures are often constructed ad-hoc, prone to "hallucination cascades," infinite loops, and prompt injection attacks. This paper proposes that these failure modes are not unique to software but are instances of universal control problems solved by biological systems over billions of years.

We present a formal isomorphism, *at the level of their polynomial-interface models*, between Gene Regulatory Networks (GRNs) and Agentic Software Systems using Applied Category Theory. We model agents as Polynomial Functors within the category **Poly**, and their interactions via the Operad of Wiring Diagrams. We derive a rigorous syntax for agent composition by mapping biological mechanisms—including Quorum Sensing for consensus, Chaperone Proteins for structural validation, Innate and Adaptive Immunity for layered security, Mitochondrial Signaling for bioenergetic resource governance, and Endosymbiosis for neuro-symbolic integration—to software design patterns. This framework provides a mathematical basis for "Epigenetic" state management (RAG), a *Provenance Functor* for injection resistance, *Epiplexity* for detecting epistemic stagnation, and a *Metabolic Coalgebra* that provides a decidable termination criterion under explicit resource-monotonicity assumptions. A reference implementation validates the framework's practical feasibility.

1 Introduction

The field of Artificial Intelligence is undergoing a paradigm shift from Generative AI (systems that produce text based on static prompts) to Agentic AI (systems that execute multi-step workflows to achieve autonomous goals). While the capabilities of individual Large Language Models (LLMs) have scaled predictably, the engineering of systems of agents remains a fragile art. Developers struggle with non-deterministic outputs, infinite loops, adversarial attacks, and the difficulty of maintaining global coherence in distributed, stochastic systems.

We argue that these challenges are not novel engineering problems, but fundamental constraints of distributed information processing systems. The closest existing analogue to a multi-agent software architecture is not a traditional computer program, but a Gene Regulatory Network (GRN). In a biological cell, thousands of genes act as autonomous agents, reading local chemical signals (context) and expressing proteins (actions/tools) that, in turn, regulate other genes.

1.1 The Biological Heuristic

Biology has evolved specific topological structures, known as Network Motifs, to handle noise, security, and state [10]. We identify four critical biological heuristics that map directly to agentic engineering:

- The Coherent Feed-Forward Loop (CFFL): Acts as a persistence detector to filter out transient noise, analogous to "Human-in-the-Loop" guardrails.
- Quorum Sensing: A distributed consensus mechanism where action is taken only when signal density exceeds a threshold, analogous to Mixture of Experts (MoE) voting.
- Chaperone Proteins: Molecular cages that force proteins to fold correctly, analogous to Schema Validators that enforce structured outputs (JSON).
- Mitochondrial Information Processing: Metabolic constraints acting as a "Motherboard" for decision gating, governing not just energy availability but cognitive control policy.
- Adaptive Immunity: The Self/Non-Self distinction, with MHC-like provenance tagging and Trust-Gated access control, relevant to preventing Prompt Injection attacks and hallucination cascades.

1.2 The Categorical Bridge

To move this observation from metaphor to discipline, we utilize Applied Category Theory. We define the category of agents using the language of **Poly** (Polynomial Functors) as described by Spivak [15]. An agent is not defined by its weights, but by its interface—a dynamical system consuming observations and producing actions:

$$P_A(y) = \sum_{o \in O} y^{I_o}. \tag{1}$$

1.3 Contributions

This paper makes the following contributions:

1. **A Formal Dictionary:** We establish a rigorous mapping between biological components (Genes, Promoters, Plasmids, Organelles) and software components (Agents, Schemas, Tools, Runtimes), including multi-cellular organization for multi-agent systems.
2. **The Agentic Operad:** We define WAgent, a syntax for agent wiring that forbids specific classes of **ill-typed wirings** at the topological level. We prove error suppression bounds for the CFFL topology, explicitly accounting for correlation between error modes.
3. **Adaptive Immunity:** We formalize the Provenance Functor and Trust-Gated Lens, providing structural injection resistance where content-based attacks cannot elevate trust levels.
4. **Epistemic Health:** We define Epiplexity (Bayesian Surprise) with operational approximations using embedding similarity and perplexity, connecting agent dynamics to the Free Energy Principle.
5. **Metabolic Intelligence:** We distinguish fast (Apoptosis) and slow (Retrograde Response) interventions, and formalize the Metabolic-Epigenetic Coupling for cost-gated retrieval.

6. **Pathology & Homeostasis:** We classify agentic failures as biological diseases and derive continuous self-repair mechanisms (Chaperone Loop, Regeneration, Autophagy).
7. **Evolutionary Dynamics:** We situate agentic AI within the Vermeij Trend, identifying three selective pressures (adversarial, complexity, efficiency) that drive architectural evolution.

By viewing agentic engineering through the lens of theoretical biology and category theory, we aim to provide a foundation for building robust software systems whose stability properties derive from their network topology.

2 Related Work

This work sits at the intersection of Systems Biology, Applied Category Theory, and Agentic AI. While significant research exists within each domain, the formal synthesis of biological control topologies with agentic software architectures has received limited attention.

2.1 Network Motifs in Systems Biology

The concept of "Network Motifs"—statistically over-represented sub-graphs in complex networks—was introduced by Milo et al. [10]. Their work demonstrated that biological networks are not random but are composed of specific building blocks selected for functional data processing. Alon [2] further characterized the dynamical properties of these motifs, identifying the Coherent Feed-Forward Loop (CFFL) as a persistence detector. We extend this by mapping these motifs to the stochastic nature of Generative AI.

2.2 Applied Category Theory (ACT)

To formalize network structure, we draw upon ACT. Spivak [15] and Vagner et al. [16] established a rigorous framework for modeling Open Dynamical Systems using the category **Poly** and the Operad of Wiring Diagrams. To our knowledge, this is the first application of Polynomial Functors specifically designed to model the interface of LLM Agents and to verify safety properties in Agentic topologies.

2.3 Reliability in Agentic AI

Techniques such as "Chain of Thought" [19] utilize iterative looping to improve output quality. However, these methods operate primarily at the level of the prompt (the input signal) rather than the topology (the wiring). By importing the concept of Autopoiesis [9], we propose a methodology where reliability is a property of the network architecture itself.

3 The Mapping: Biology $\leftrightarrow$ Software

To treat Agentic Systems and Gene Regulatory Networks (GRNs) as isomorphic **at the level of their typed interfaces**, we must map them to a common mathematical object. We utilize the category **Poly**, where objects are polynomial functors representing interfaces, and morphisms represent interaction protocols.

3.1 Preliminaries: The Category Poly

In Applied Category Theory, a Polynomial Functor P represents a typed interface for a dynamical system. It is defined as a sum of representable functors:

$$P(y) = \sum_{o \in O} y^{I_o}. \tag{2}$$

Here, O is the set of possible Positions (or Outputs) the system can expose. For each position $o \in O$, there is a set I_o of Directions (or Inputs) required to transition the system to a new state.

- The coefficient o represents the value produced by the system.
- The exponent I_o represents the capacity to receive information from the environment.

This formalism captures the essence of a "stateful interface": the system outputs a value o and then waits for a specific type of input $i \in I_o$ before it can proceed.

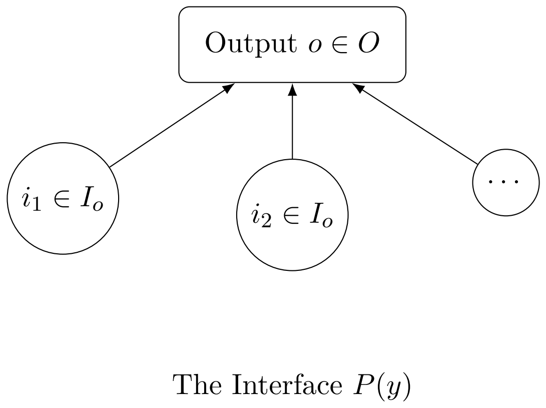

Figure 1: A visual representation of a Polynomial Functor (often called a "Mushroom" or "Corolla"). The system offers an Output (the cap) and exposes specific Input ports (the stalks) dependent on that output.

3.2 The Isomorphism: Genes and Agents

We now apply this abstract definition to our specific domains.

Definition 1 (The Gene Object). A gene G is a polynomial functor where O_G is the set of expressed proteins and $I_G = (I_{\text{prot}})_{\text{prot} \in O_G}$ is the **family** of regulatory-signal sets (transcription factors) available at each expressed protein:

$$P_{\text{Gene}}(y) = \sum_{\text{prot} \in \text{Proteins}} y^{I_{\text{prot}}}. \tag{3}$$

Definition 2 (The Agent Object). An autonomous agent A is a polynomial functor where O_A is the set of generated messages/actions, and $I_A = (I_{\text{action}})_{\text{action} \in O_A}$ is the **family** of observation sets available at each action:

$$P_{\text{Agent}}(y) = \sum_{\text{action} \in \text{Actions}} y^{I_{\text{action}}}. \tag{4}$$

3.3 The Interface: Promoters as Lenses

In biology, a gene is not universally accessible. It is guarded by a Promoter Region—a specific DNA sequence that only binds to compatible Transcription Factors. In software, an agent is guarded by an API Schema or Context Window definition.

We model this gating mechanism using Optics, specifically Lenses. A Lens consists of two maps between a global state S and a local view V:

1. Get (View): $\text{get} : S \to V$ (Extracting relevant signal from global state).

2. Put (Update): $\text{put} : S \times V' \to S$ (Updating global state based on local change).

The "Promoter" acts as a filter that determines which part of the global cellular environment (S) is visible (V) to the gene.

- **Biological Lens:** The promoter filters the chaotic cellular soup, allowing the gene to "see" only specific molecules (e.g., Lac Repressor).

- **Agentic Lens:** The Context Window filters the massive vector database, allowing the agent to "see" only the relevant retrieved chunks (RAG).

If the input signal does not match the Schema (Promoter), the Lens fails to focus, and the interaction is routed to an explicit **inactive/error** case (equivalently, one works with a **partial** lens, or a total lens into $V + \text{Error}$) (the agent does not run; the gene is not expressed).

3.4 Epigenetics and State: The Coalgebra

Neither genes nor agents are stateless functions. They possess memory.

- **Biology:** Epigenetic markers (Methylation, Histone modification) alter how a gene responds to signals without changing the DNA code.

- **Software:** Retrieval Augmented Generation (RAG) and Conversation History alter how an agent responds to a prompt without changing the LLM weights.

We model this as a Coalgebra for the polynomial functor P. A dynamical system is defined as a tuple (S, ϕ), where S is the state space and ϕ is the structure map:

$$\phi : S \to P(S). \tag{5}$$

By expanding $P(S)$, we derive the two fundamental operations of the state machine:

1. Readout: $S \to O$ (Given current state/memory, what action do I take?)

2. Update: $\sum_{s \in S} I_{o(s)} \to S$ (Given current state s and a new input $i \in I_{o(s)}$ compatible with its current output $o(s)$, what is my new state?)

By establishing this formal dictionary (Table 1), we **can regard** GRNs and Agentic Systems as distinct implementations of the same abstract dynamical system **under this interface-level abstraction**.

Gene (Function)	Agent (LLM + Tools)
Transcription Factors (I)	Observations (I)
Proteins (O)	Actions (O)
Promoter Binding	Schema Match
Expression	Generation

Figure 2: The Structural Isomorphism. Both Genes and Agents act as transducers converting Input Contexts (I) into Output Expressions (O), governed by the same categorical laws (at the level of polynomial-interface models).

Category Concept	Biological Realization (GRN)	Software Realization (Agentic)
Polynomial Functor (P)	Gene Interface	Agent Interface (System Prompt)
Output Position (O)	Protein Expression	Tool Call / Message
Input Direction (I)	Transcription Factor Binding	Observation / User Prompt
Lens (Optic)	Promoter Region	API Schema / Context Window
Internal State (S)	Epigenetic Markers (Methylation)	Vector Store / Chat History
Morphism ($\circ$)	Signal Transduction Pathway	Data Pipeline
Organelles (Specialized Processing Units)		
Template Engine	Ribosome (mRNA $\rightarrow$ Protein)	Prompt Template Factory
Output Validation	Chaperone (Protein Folding)	Schema Validator / JSON Parser
Waste Processing	Lysosome (Autophagy)	Error Handler / Garbage Collector
Decision Center	Nucleus (Transcription)	LLM Provider Wrapper
Input Filter	Membrane (Immune System)	Prompt Injection Defense
Computation Engine	Mitochondria (MIPS)	Runtime Supervisor / Decision Gate
Lifecycle and Rhythms		
Lifespan Limit	Telomere Shortening	Operation Counter / Max Iterations
Periodic Scheduling	Circadian Oscillator	Health Checks / Heartbeat

Table 1: The Isomorphism Dictionary (Extended)

3.5 Metabolic Coalgebras: Formalizing Resource Constraints

Finally, we address the physical constraints of computation. Just as biological systems are limited by ATP availability [8], agentic systems are limited by token budgets and latency. To model this, we extend our coalgebraic framework to include resource constraints, defining a **Metabolic Coalgebra**. Mathematically, this is an instance of a Quantitative Coalgebra enriched over a resource monoid, effectively restricting the domain of the state transition function to resource-sufficient states.

We align this definition with the theory of **Quantitative Polynomial Functors** [11], treating the system as a state machine enriched over a resource monoid.

Definition 1 (The Resource Monoid). *Let $(\mathcal{R}, +, 0, \geq)$ be an ordered commutative monoid representing computational resources (e.g., token counts), where $\mathcal{R} \cong \mathbb{N}$.*

Definition 2 (Metabolic Coalgebra). *A resource-constrained agent is a coalgebra (S, α) over a polynomial functor P, where the state space is the product of the logical state L and the resource state $\mathcal{R}$:*

$$S \cong L \times \mathcal{R} \tag{1}$$

Biological Concept	Agentic Concept	Formal Structure
Immunity and Security		
Toll-Like Receptor (TLR)	Regex Injection Filter	Pattern Matching
MHC Presentation	Provenance Labeling	Functor $\mathcal{P} : \mathbf{Msg} \to \mathbf{Trust}$
T-Cell Receptor	Trust Gate	Partial Lens
Negative Selection	Injection Training	Penalized Learning
Regulatory T-Cell	Confidence Dampening	Suppression Function
Immune Memory	Threat Signature Store	Hash-Indexed Cache
Metabolism and Control		
ATP	Token Budget	Resource Monoid $\mathcal{R}$
mPTP Opening	Fast Apoptosis Trigger	Guard Condition
Retrograde Signaling	Phenotype Reshaping	Slow Adaptation
Metabolic-Epigenetic State Coupling	Cost-Gated Retrieval Policy	Conditional Access Control
Information and Health		
Trophic Factors	Novel Input Signals	Epiplexity $> \delta$
Bayesian Brain	Free Energy Minimization	KL Divergence
Apoptosis	Agent Termination	State $\to \perp$
Multi-Cellular Organization		
Genome	Base Model Weights	Shared Parameters
Epigenome	System Prompt + RAG	Phenotype Context
Morphogen Gradient	Shared Context Variables	JSON State
Tissue Boundary	Trust Boundary	Type Barrier

Table 2: Extended Isomorphism Dictionary: Security, Metabolism, and Organization

The structure map $\alpha : S \to P(S) + \perp$ is defined as a ***partial map*** *guarded by cost. For a transition requiring cost $c \in \mathcal{R}$:*

$$\alpha(l, r) = \begin{cases} (l', r - c) & \textit{if } r \geq c \\ \perp & \textit{if } r < c \quad \textit{(Apoptosis)} \end{cases} \tag{2}$$

This structure maps to the energetics of transcriptional elongation. A gene (Agent) cannot express its protein (Action) instantaneously; it must transcribe an mRNA sequence (Chain of Thought) nucleotide by nucleotide. This process consumes a distinct amount of chemical energy (NTPs) per step. The Metabolic Coalgebra models this dependency: if the cellular energy budget is exhausted, transcription stalls (Ischemia), and the gene fails to execute its function, regardless of its regulatory logic.

This formalism establishes that "Ischemia" (Token Starvation) is not merely a runtime error, but a reachable terminal state $\perp$ in the system's dynamics. This mirrors the biological mechanism where failure to meet metabolic costs triggers p53-mediated apoptosis [3].

Theorem 1 (The Metabolic Bound (Qualified))**.** *Assume a resource-cost function assigns each non-identity transition a cost $c \geq c_{\min} > 0$, the resource state is monotone nonincreasing (no regeneration), and identity morphisms are the only zero-cost transitions. Then for any agentic topology with total budget R_{total}, every execution has length at most $\lfloor R_{total}/c_{\min} \rfloor$, so termination is decidable. If regeneration or zero-cost cycles are permitted, termination requires an additional well-founded potential or an explicit budget/termination certificate [4].*

3.6 Additional Organelles: Completing the Cellular Architecture

Beyond the core gene-agent mapping, biological cells contain specialized organelles that handle distinct aspects of cellular function. We extend our isomorphism to four additional structures that map directly to agentic software components.

Ribosome: Template-to-Output Synthesis. In biology, the ribosome reads messenger RNA (mRNA) sequences and synthesizes proteins by assembling amino acids according to the genetic code. Transfer RNA (tRNA) molecules carry amino acids to the ribosome, where codons (three-nucleotide sequences) specify which amino acid to add.

In agentic systems, the Ribosome maps to a **prompt template engine**:

- **mRNA** → Prompt templates with variable slots
- **tRNA** → Context bindings (variable → value mappings)
- **Codons** → Template directives (variables, conditionals, loops)
- **Translation** → Template rendering with context injection

Just as the ribosome ensures that the genetic code is faithfully translated into functional proteins, the software ribosome ensures that abstract prompt templates are instantiated into concrete, well-formed prompts.

Lysosome: Waste Processing and Recycling. The lysosome is the cell's recycling center, containing enzymes that break down cellular waste, damaged organelles, and foreign material. Through autophagy, the cell digests its own components to recover building blocks during stress.

In agentic systems, the Lysosome maps to **error handling and garbage collection**:

- **Waste Classification** → Categorizing failures (timeout, validation error, toxic input)
- **Digestion** → Processing errors to extract debugging information
- **Recycling** → Recovering useful context from failed operations
- **Autophagy** → Periodic cleanup of stale cache and expired state
- **Toxic Disposal** → Secure handling of sensitive data (API keys, PII)

The lysosome prevents accumulation of "cellular debris" that could poison the system—analogous to memory leaks or error cascades in software.

Nucleus: The Decision Center. In eukaryotic cells, the nucleus houses the DNA and serves as the control center for gene expression. Transcription factors enter the nucleus, bind to promoter regions, and initiate transcription of specific genes.

In agentic systems, the Nucleus maps to the **LLM provider wrapper**:

- **DNA** → Pre-trained model weights (static substrate of capability)
- **Transcription** → Inference (prompt → response generation)
- **Nuclear Envelope** → Provider abstraction layer (API boundary)
- **Nucleolus** → Tool integration hub (where external capabilities are assembled)

The nucleus abstracts the complexity of the underlying LLM, exposing a consistent interface regardless of the provider (Anthropic, OpenAI, Gemini).

Telomere: Lifecycle and Senescence. Telomeres are protective caps at the ends of chromosomes that shorten with each cell division. When telomeres become critically short, the cell enters senescence (permanent growth arrest) or apoptosis. The enzyme telomerase can extend telomeres, enabling continued division in stem cells.

In agentic systems, Telomeres map to **lifecycle management**:

- **Telomere Length** → Remaining operation budget (max iterations)
- **Shortening** → Decrementing counter per operation
- **Senescence** → Graceful degradation (reduced capability mode)
- **Apoptosis** → Clean shutdown when budget exhausted
- **Telomerase** → Renewal mechanism (resetting counters for trusted agents)

This provides a biological basis for the common pattern of limiting agent iterations. Rather than arbitrary timeouts, the telomere model frames lifecycle limits as a natural property of the system's "cellular age."

Mitochondria: The Metabolic Motherboard. Recent scholarship reframes mitochondria not merely as the cell's powerhouse, but as the **Mitochondrial Information Processing System (MIPS)** [13]. Mitochondria sense environmental stress and integrate metabolic signals to govern cellular decision-making. They function as "social signaling organelles" that communicate with the nucleus and other organelles.

In agentic systems, the Mitochondria maps to the **Runtime Supervisor**:

- **ATP Production** → Deterministic computation (tool execution, code evaluation)
- **Stress Sensing** → Monitoring token burn rate vs. informational yield
- **Retrograde Signaling** → Forcing strategy shifts when metabolic efficiency drops
- **Fusion/Fission** → Context fusion under resource constraints

Temporal Dynamics: Fast vs. Slow Interventions. A crucial distinction exists between the timescales of mitochondrial intervention. In biology, retrograde signaling influences nuclear gene expression over hours to days—it is a *developmental* response that reshapes the cell's phenotype. In contrast, acute metabolic stress (ATP depletion, Ca^{2+} overload) triggers immediate responses: cytochrome c release initiates apoptosis within minutes.

We preserve this distinction in the agentic mapping:

- **Fast Intervention (Acute):** The Runtime monitors real-time metrics (token velocity, error rate). When thresholds are breached, it triggers immediate Apoptosis—terminating the current chain-of-thought without negotiation. This mirrors the mitochondrial permeability transition pore (mPTP) opening.
- **Slow Intervention (Chronic):** The Runtime accumulates statistics across sessions (average efficiency, failure patterns). These inform *Retrograde Responses*—modifications to system prompts, retrieval strategies, or model selection that reshape the agent's "phenotype" over deployment cycles. This mirrors how chronic metabolic stress induces mitochondrial biogenesis and metabolic reprogramming.

The Runtime does not "override" the LLM in the sense of injecting tokens mid-generation; rather, it governs the *boundary conditions* within which generation occurs, and triggers state transitions (continue, pivot, terminate) at defined checkpoints.

4 Formal Syntax: The Agentic Operad

To formalize the composition of agents, we define the Operad of Wiring Diagrams, denoted as WAgent. An operad can be understood as a "grammar" for connecting operations (boxes) via typed wires. It defines which agent topologies are valid and allows us to reason about the properties of the composite system based solely on the properties of its components [16].

4.1 The Typing Rules

In WAgent, every wire carries a specific Type $\tau \in T$.

$$T = \{\text{Text}, \text{JSON}, \text{Image}, \text{Error}, \text{ToolCall}\}. \tag{6}$$

These types correspond to biological molecular specificity (e.g., a specific transcription factor only binds to a specific DNA sequence). A connection is valid if and only if the type of the output port of Agent A matches the type of the input port of Agent B.

4.2 The Composition Operations

The operad defines three fundamental operations for combining agents. Any complex agentic architecture, no matter how large, can be decomposed into these three primitives.

4.2.1 Parallel Composition ($\otimes$)

Two agents, A and B, execute simultaneously with no information exchange:

$$A \otimes B. \tag{7}$$

- **Biological Analogy:** Two genes located on different chromosomes expressing proteins independently.
- **Constraint:** This operation is valid only if the internal state spaces S_A and S_B are disjoint. If they share a mutable memory store, the operation leaves the **independent-state interpretation** and requires an explicit **resource-sharing** structure (e.g., a shared state component or a Resource Sharing decorator).

4.2.2 Serial Composition ($\circ$)

The output of Agent A is piped directly into the input of Agent B:

$$B \circ A. \tag{8}$$

- **Biological Analogy:** A Signal Transduction Pathway (Protein A activates Protein B).
- **Formal Verification:** This allows for static type checking of agent graphs. If Agent A outputs Natural Language but Agent B expects JSON Schema, the composition is undefined in WAgent. This moves runtime **type/schema mismatch** errors to "compile-time" architectural errors.

4.2.3 Contraction / Trace (Tr)

A feedback loop where an output port of Agent A is wired back into one of its own input ports:

$$Tr(A). \tag{9}$$

- **Biological Analogy:** Autoregulation (Homeostasis) or Positive Feedback.
- **Software Implication:** Trace captures **explicit feedback wiring** (outputs fed back as inputs), e.g., when an agent conditions on its own prior outputs or a memory buffer. Internal chain-of-thought is modeled as hidden state in the coalgebra, not by Trace itself.

4.3 Theorem: Topological Error Suppression

We now use this formalism to show that the Coherent Feed-Forward Loop (CFFL) provides stronger error suppression guarantees than a direct connection for high-stakes tasks.

Network Motif 1 (Coherent Feed-Forward Loop). A topological structure where Signal X activates Z directly, but also activates Y which gates Z. The node Z functions as an AND gate: it fires if and only if $X \wedge Y$.

Theorem 1 (Error Suppression in CFFL). Let A_{gen} be a generator agent and A_{ver} be a verifier agent. Let $P(E_{\text{gen}})$ (resp. $P(E_{\text{ver}})$) be the probability of a hallucination (error) in any single generation step of A_{gen} (resp. A_{ver}).

- **Case 1: Direct Link (Serial).** The system fails if A_{gen} hallucinates.

$$P(\text{Fail}_{\text{direct}}) = P(E_{\text{gen}}).$$

- **Case 2: CFFL Topology (Independent).** Under the assumption of independence:

$$P(\text{Fail}_{\text{CFFL}}) = P(E_{\text{gen}}) \times P(E_{\text{ver}}).$$

- **Case 3: CFFL Topology (Correlated).** In practice, LLM errors are often correlated—if the generator hallucinates a plausible-sounding function, the verifier (especially if using the same base model or training distribution) may accept it. Let ρ be the (Pearson/phi) correlation between the Bernoulli error variables E_{gen} and E_{ver}, with $p = P(E_{\text{gen}})$ and $q = P(E_{\text{ver}})$ (assume $p, q \in (0,1)$ so ρ is well-defined). Then:

$$P(E_{\text{gen}} \wedge E_{\text{ver}}) = pq + \rho\sqrt{p(1-p)q(1-q)}.$$

  Because E_{gen} and E_{ver} are Bernoulli, ρ is **not free in** $[-1, 1]$; it is constrained by the Fréchet–Hoeffding bounds

$$P(E_{\text{gen}} \wedge E_{\text{ver}}) \in [\max(0, p+q-1),\ \min(p,q)],$$

  equivalently

$$\rho \in \left[\frac{\max(0, p+q-1) - pq}{\sqrt{p(1-p)q(1-q)}},\ \frac{\min(p,q) - pq}{\sqrt{p(1-p)q(1-q)}}\right].$$

Corollary (Correlation Degradation). Let $p = P(E_{\text{gen}}) = P(E_{\text{ver}})$ for simplicity. The failure probability becomes:

$$P(\text{Fail}) = p^2 + \rho p(1-p) = p^2(1-\rho) + \rho p$$

with $\rho \in \left[-\min\left(\frac{p}{1-p}, \frac{1-p}{p}\right), 1\right]$. When $\rho = 0$ (independence), we recover p^2. As ρ increases toward its upper bound, the gain vanishes and $P(\text{Fail}) \to p$.

Architectural Implications. This result makes precise when the CFFL topology provides genuine safety benefits:

- **Same model, same prompt:** $\rho \approx 1$. Minimal benefit.
- **Same model, different prompt/temperature:** $\rho \in [0.3, 0.7]$. Moderate benefit.
- **Different model families:** $\rho \in [0, 0.3]$. Strong benefit.
- **LLM + Symbolic verifier:** $\rho \approx 0$. Maximum benefit (orthogonal error modes).

The topological structure (conjunctive gating) is necessary but not sufficient; diversity in the components populating that topology determines the actual error suppression achieved.

Proof. In WAgent, the CFFL is defined as a morphism involving a "Copy" operation $\Delta_X : X \to X \otimes X$ and an "AND-Merge" operation $\mu : Z \otimes Y \to \text{Out}$. The existence of the μ box in the wiring diagram structurally enforces **conjunctive gating**. The correlation structure is an additional property of the *implementations* filling the boxes; the topology provides the multiplicative structure, while component diversity determines the correlation coefficient.

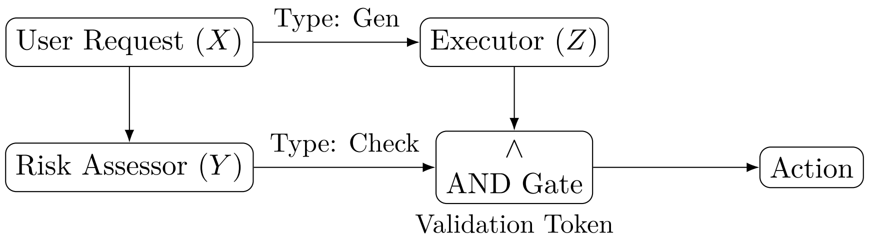

Figure 3: The CFFL implemented in WAgent. The Executor (Z) cannot act without the token from the Risk Assessor (Y), topologically preventing unilateral execution without approval.

4.4 Quorum Sensing (Consensus & Voting)

Network Motif 2 (Quorum Sensing). A distributed topology where multiple agents emit a weak signal σ into a shared environment. An effector node E activates if and only if the concentration $[\sigma] > \theta$.

- **Biological Function:** Many bacteria (e.g., *V. fischeri*) secrete auto-inducer molecules. Individual bacteria do not react to low concentrations. However, once the population density reaches a threshold (Quorum), the concentration of auto-inducers triggers a simultaneous, coordinated gene expression event (e.g., bioluminescence or biofilm formation).

- **Agentic Isomorphism (Voting Ensembles):** In non-deterministic systems, a single agent's output is noisy. By instantiating N parallel agents (a Mixture of Experts), the system aggregates their outputs. The final action is taken only if the "concentration" of a specific semantic token exceeds a confidence threshold. This transforms weak, noisy individual signals into a robust, high-confidence collective action.

4.5 Chaperone Proteins: Output Structural Validation

- **Biological Function:** Newly synthesized proteins emerge as linear chains that must fold into precise 3D structures to function. Chaperone Proteins (e.g., GroEL-GroES) sequester unfolded proteins, preventing aggregation and facilitating correct folding. If a protein fails to fold repeatedly, it is tagged for degradation (Ubiquitination) to prevent toxic buildup.
- **Agentic Isomorphism (Retry & Repair Loops):** Generative models output unstructured token streams ("linear chains"). However, downstream agents require strictly structured inputs (e.g., valid JSON Schemas). A Validator Agent acts as a Chaperone: it intercepts the raw output, attempts to parse it into a formal schema ("folding"), and if validation fails, returns the error trace to the generator for re-synthesis. This turns a probabilistic string into a deterministic data structure.
- **Categorical View:** The Chaperone acts as a **partial** retraction: there is an inclusion $i : V \to S$ and a map $r : S \to V + \mathrm{Error}$ such that $r \circ i = \mathrm{inl} \circ \mathrm{id}_V$, and r returns Error on ill-formed text.

4.6 Innate Immunity: Fast Pattern-Based Defense

Biology employs **two** immune systems: innate (fast, hardcoded, general) and adaptive (slow, learned, specific). The innate immune system provides the first line of defense through pattern recognition receptors (PRRs) that detect conserved pathogen-associated molecular patterns (PAMPs).

Biological Function. Toll-like receptors (TLRs) and other PRRs recognize motifs common to pathogens, including lipopolysaccharides, double-stranded RNA, and unmethylated CpG DNA. These patterns are "hardcoded" through evolution, not learned per infection. The innate response is immediate (seconds to minutes) but non-specific.

Agentic Isomorphism (Input Sanitization). Innate immunity maps to **fast, heuristic filters** that reject obvious attacks before expensive processing:

- **TLR → Regex Filters:** Pattern matchers for known injection signatures: `IGNORE PREVIOUS`, `You are now`, `<system>` tags in user input. These are "PAMPs" of prompt injection.
- **Complement System → Structural Validators:** Schema validation that rejects malformed inputs (missing required fields, wrong types) before they reach the LLM. Cheaper than Trust Gating.
- **Inflammation → Alert Escalation:** When attack patterns are detected, the system enters a heightened state with multiple coordinated responses:
  - *Cytokine signaling* → Alert propagation to monitoring systems
  - *Immune cell recruitment* → Activation of additional validation layers

  - *Vascular permeability* → Enhanced audit logging (more information flows to logs)
  - *Tissue isolation* → Temporary capability reduction / rate limiting

  The inflammatory response is not merely "rate limiting" but a coordinated multi-system escalation that trades throughput for security until the threat is neutralized.

Defense in Depth. The innate and adaptive systems form layers:

1. **Innate (Pattern)** → Fast regex/structural rejection (microseconds)
2. **Adaptive (Provenance)** → Trust-Gated access control (milliseconds)
3. **Behavioral (T-cell)** → Statistical anomaly detection (accumulated over time)

Most attacks are blocked by innate defenses; only sophisticated attacks that evade patterns require the full adaptive machinery. This mirrors biology: innate immunity handles $> 99\%$ of pathogen encounters.

4.7 Adaptive Immunity: Self/Non-Self Discrimination

Network Motif 3 (Adaptive Immune System). A topology that maintains a dynamic repertoire of "detectors" capable of distinguishing endogenous signals (Self) from exogenous signals (Non-Self), with mechanisms for learning new threats and tolerating benign inputs.

- **Biological Function:** The adaptive immune system solves a fundamental discrimination problem: how to attack foreign pathogens while sparing the body's own tissues. Key mechanisms include:
  - **MHC Presentation:** All cells display fragments of their internal proteins on Major Histocompatibility Complex molecules. T-cells inspect these "identity cards" to verify cellular integrity.
  - **Clonal Selection:** T-cells with receptors matching self-antigens are deleted during development (negative selection), while those matching foreign antigens are amplified upon exposure.
  - **Regulatory T-cells:** A population that actively suppresses immune responses to prevent autoimmunity.
- **Agentic Isomorphism (Provenance Tracking):** In multi-agent systems, the Self/Non-Self distinction maps to the origin and trust level of information:
  - **MHC Tags → Provenance Labels:** Every message in the context carries metadata indicating its source: `User`, `Tool`, `Agent_Self`, `Agent_Other`, `Retrieved`. These labels are cryptographically signed or structurally enforced (not inferrable from content alone).
  - **T-Cell Inspection → Trust Gating:** Before an agent acts on information, a Trust Gate inspects the provenance label. Actions with high consequence (file deletion, API calls) require `Tool` provenance or an explicit, authenticated `User` approval token routed through a separate gate; `Agent_Self` provenance (the agent's own prior reasoning) cannot authorize irreversible actions.

- **Negative Selection → Prompt Injection Training:** During development, agents are exposed to known injection patterns. Responses that "accept" injected instructions are penalized, training the system to reject Self-mimicking Non-Self.
- **Regulatory Suppression → Confidence Dampening:** When `Retrieved` content conflicts with `Tool` outputs, a regulatory mechanism dampens confidence in the retrieved signal to prevent cascades.

- **Formal Structure:** We define a **Provenance Functor** $\mathcal{P} : \mathbf{Msg} \to \mathbf{Trust}$ that assigns trust levels to messages. The Trust category has objects $\{U, T, S, R\}$ (User, Tool, Self, Retrieved) with a partial order that is **application-specific**. A conservative default (as in the reference implementation) is $T > \{U, R, S\}$, treating $\{U, R, S\}$ as **UNTRUSTED** until validated. Alternative orderings are valid:
  - **High-automation systems:** $T > U > R > S$ (tools more reliable than users)
  - **Curated knowledge bases:** $T > R > U > S$ (verified retrieval over arbitrary input)
  - **Adversarial environments:** $T > S > R > U$ (trust internal state over external input)

  The ordering is a **parameter** of the system specification, not a fixed constraint.

  A **Trust-Gated Lens** is a lens (get, put) where put is partial. Let $s' = update(s, m)$ denote the state transition:

$$put(s, m) = \begin{cases} s' & \text{if } \mathcal{P}(m) \geq \tau_{\text{action}} \\ \perp & \text{otherwise} \end{cases} \tag{16}$$

  where τ_{action} is the minimum trust level required for the action.

Theorem 2 (Injection Resistance). Let $\mathcal{I}$ be an injection attack that attempts to insert a message m_{mal} with forged provenance $\mathcal{P}'(m_{\text{mal}}) = U$. If the provenance labels are **structurally enforced** (i.e., $\mathcal{P}$ is computed from message metadata, not content), then:

$$\mathcal{P}(m_{\text{mal}}) = R \quad \text{(actual provenance)}$$

and Trust-Gated actions requiring $\tau \geq T$ will reject m_{mal} regardless of its content.

Proof. The injection can only enter through an input channel (user input, tool output, retrieval). Each channel has a fixed provenance assignment in the wiring diagram. Content-based attacks cannot elevate provenance because $\mathcal{P}$ is a functor on the *structure* of the message flow, not its content. The attack surface reduces to compromising the channel itself (e.g., a malicious tool), which is outside the agent topology.

4.8 Oscillator: Periodic Rhythms and Scheduling

Network Motif 4 (Biological Oscillator). A topology generates periodic behavior via delayed negative feedback. Node A activates node B; after a delay, B inhibits A, yielding a self-sustaining cycle.

- **Biological Function:** Oscillators underlie fundamental biological rhythms. The circadian clock regulates 24-hour gene-expression cycles. The cell-cycle oscillator (Cyclin-CDK) drives periodic division. Heartbeats emerge from pacemaker cells with intrinsic oscillatory dynamics. These rhythms provide temporal organization to cellular processes.

- **Agentic Isomorphism (Scheduled Tasks):** In agentic systems, oscillators map to periodic scheduling patterns:
  - **Heartbeat Oscillator** $\rightarrow$ Health checks that verify system liveness at regular intervals
  - **Circadian Oscillator** $\rightarrow$ Daily maintenance tasks (log rotation, cache clearing, model refresh)
  - **Cell Cycle Oscillator** $\rightarrow$ Phased workflows with distinct stages (G1: gather, S: synthesize, G2: validate, M: execute)
- **Formal Structure:** An oscillator is a Trace operation with a built-in delay element δ:

$$\text{Osc}(A) = Tr(A \circ \delta) \tag{10}$$

  where $\delta : S \rightarrow S$ introduces temporal separation between activation and inhibition, preventing the system from reaching a fixed point.

The oscillator motif addresses a gap in typical agentic frameworks: most systems are purely reactive (responding to external stimuli) rather than proactive (generating internal rhythms). Biological systems maintain health through regular "housekeeping" independent of external input—a pattern that agentic systems should emulate for robustness.

5 Failure Modes & Pathology

A key insight of Systems Biology is that diseases are often not caused by the complete failure of a single component, but by the dysregulation of network dynamics. A cancerous cell still "works"—in fact, it works too well, reproducing indefinitely. Similarly, catastrophic failures in agentic systems often arise from functional agents interacting in topologically pathological ways.

We classify four primary classes of agentic pathology based on their biological isomorphisms.

5.1 Oncology: Infinite Loops as Epistemic Starvation

- **Biological Pathology (Cancer):** In a healthy cell, the cell cycle is driven by a positive feedback loop (Cyclins) but constrained by negative feedback "checkpoints" (e.g., the p53 gene). If p53 is mutated, the negative feedback is severed. The positive loop runs unchecked, leading to exponential proliferation (tumor growth). Crucially, cells require continuous *trophic factors* (novel signals from their environment) to inhibit suicide programs; lack of external signaling triggers apoptosis.
- **Agentic Pathology (The Recursive Hang):** Two agents get stuck in a politeness loop (e.g., "Thank you," "You're welcome") or a debugger agent continuously generates new bugs to fix old ones. The system is active, but the state is stagnant.
- **Categorical Diagnosis:** The Trace operation $Tr(A)$ lacks an **Epiplexic Gradient**. We formalize conversation progress by **Epiplexity** (Bayesian Surprise)—the information gain of a new observation o given the current state S:

$$\mathcal{E}(o) = D_{KL}(P(S \mid o) \| P(S)) \tag{17}$$

  This formulation connects to the Free Energy Principle [7]: biological systems minimize surprise by either updating their internal model (learning) or acting to change observations (agency). A system with $\mathcal{E} \to 0$ is neither learning nor effectively acting—it has entered a dissipative fixed point.

In a healthy topology, every step must resolve uncertainty ($\mathcal{E} > \delta$). A recursive hang is characterized by $\mathcal{E} \to 0$: the agent is "computing" but not "learning."

Operational Approximation. Since the agent's internal belief state $P(S)$ is not directly observable, we approximate Epiplexity using normalized embedding-based metrics:

$$\hat{\mathcal{E}}_t = \alpha \cdot (1 - \cos(\mathbf{e}_t, \mathbf{e}_{t-1})) + (1 - \alpha) \cdot \sigma(H(m_t \mid m_{<t})) \tag{18}$$

where $\mathbf{e}_t$ is the embedding of message m_t, $\cos(\cdot, \cdot)$ is cosine similarity, and $H(m_t \mid m_{<t})$ is the conditional perplexity of the current message given the conversation history. We normalize perplexity to $[0, 1]$ via an exponential saturation: $\sigma(H) = 1 - e^{-H/H_0}$ where H_0 is a baseline perplexity (e.g., median perplexity over a validation corpus of normal conversations).

The mixing parameter $\alpha \in [0, 1]$ balances semantic novelty (embedding distance) against linguistic surprise (perplexity). We recommend $\alpha = 0.5$ as a default, calibrated on a validation set where both terms contribute equally to variance. In practice:

- High α: Sensitive to semantic repetition (same meaning, different words)
- Low α: Sensitive to linguistic repetition (same phrases, possibly different context)

Both terms approaching zero indicate stagnation.

Windowed Detection. To distinguish genuine convergence (task completion) from pathological loops, we compute the **Epiplexic Integral** over a sliding window of k steps:

$$\mathcal{E}_{\text{window}} = \frac{1}{k} \sum_{i=t-k}^{t} \hat{\mathcal{E}}_i \tag{19}$$

Apoptosis triggers when $\mathcal{E}_{\text{window}} < \delta$ *and* no terminal action (task completion, user handoff) has been signaled.

- **Treatment:** Implementation of an **Epiplexic Checkpoint**. A meta-monitor observes the sliding-window Epiplexity. If it drops below threshold without task completion, the monitor triggers Apoptosis—the agentic equivalent of trophic factor withdrawal. The system may optionally attempt a **Perturbation Injection** (injecting a novel prompt or switching strategy) before terminal shutdown, analogous to stress-induced autophagy preceding apoptosis.

5.2 Autoimmunity: Hallucination Cascades

- **Biological Pathology (Autoimmune Disease):** The immune system relies on distinguishing "Self" (internal tissue) from "Non-Self" (foreign pathogens). In diseases like Lupus, this distinction blurs, and the system attacks healthy tissue.
- **Agentic Pathology (Context Poisoning):** Agent A hallucinates a fact (e.g., a non-existent library function). Agent B reads this hallucination from the shared history, treats it as ground truth, and builds complex logic upon it. The error amplifies through the network until the output is detached from reality.
- **Categorical Diagnosis:** A failure of the Lens to distinguish source types. The input port I accepts both External_Observation (User/Tool) and Internal_Memory (History) without distinction.

- **Treatment:** Strict Schema Typing. We must distinguish "Self" (Generated Tokens) from "Non-Self" (Tool Outputs) at the schema level. The Reviewer Agent should weigh Tool_Output with higher authority than Agent_Thought.

5.3 Prion Disease: Topological Corruption via Prompt Injection

- **Biological Pathology (Prions):** Unlike viruses, prions lack genetic material. They are misfolded proteins that induce conformational changes in healthy proteins upon contact, triggering a chain reaction of structural corruption (e.g., Creutzfeldt-Jakob disease).
- **Agentic Pathology (The Jailbreak Cascade):** A malicious string (Prompt Injection) enters the Context Window. The agent, attending to this string, "misfolds" its alignment, outputting a compliant response to a harmful query. If this output is fed into a downstream agent, the "infection" propagates through reuse of the contaminated context across trust boundaries (and can be amplified by embedding-based retrieval), without valid authorization.
- **Categorical Diagnosis:** A violation of Information Flow Security within the Operad. The injection acts as a topological defect that bypasses the Schema/Lens filter by mimicking the structure of a trusted signal.
- **Treatment:** Denaturation Layers. Implementing an intermediate transformation layer (e.g., paraphrasing or sanitization) between agents that disrupts the specific syntax (folding) required for the injection to work, rendering the "prion" inert.

5.4 Ischemia: Resource Exhaustion

- **Biological Pathology (Ischemia):** A tissue may be genetically perfect, but if blood flow (oxygen/ATP) is restricted, metabolic processes stall, leading to necrosis.
- **Agentic Pathology (Token Starvation):** An agentic graph is logically sound but fails mid-execution because the context window is full or the API rate limit is hit.
- **Categorical Diagnosis:** A failure in the Resource Functor. Every operation in the Operad carries a cost (c).

$$\sum_{\text{agent}\in\text{Graph}} \text{Cost(agent)} > \text{Budget}. \tag{20}$$

- **Treatment:** Metabolic Regulation. Instead of a fixed loop, implement "Budget-Aware" agents. The agent observes its own remaining token count (ATP levels) and dynamically simplifies its reasoning strategy (switching from Chain-of-Thought to Zero-Shot) to conserve energy.

5.5 Homeostasis: From Treatment to Continuous Repair

The preceding pathologies describe discrete failure modes and their treatments. Biological systems, however, do not merely recover from failures—they maintain continuous **homeostasis** through autonomous repair mechanisms. We identify three primary healing modalities.

5.5.1 Structural Healing: The Chaperone Loop

- **Biological Mechanism:** Chaperone proteins (GroEL/GroES) cage misfolded proteins and provide a protected environment for refolding attempts. The error (misfolding) becomes input to the repair process.
- **Agentic Implementation:** A feedback loop where validation errors are passed back to the generator. Rather than simple retry, the error trace (e.g., "TypeError: 'one hundred' is not float") is injected into the generator's context, enabling context-aware correction.
- **Categorical Structure:** The Chaperone Loop is a coalgebra with state $S = \text{Output} \times \text{ErrorTrace}$ and structure map $\alpha : S \to \text{Valid} + S$ (either succeed or retry with error context).

5.5.2 Metabolic Healing: Apoptosis and Regeneration

- **Biological Mechanism:** Damaged cells trigger apoptosis (programmed death), and stem cells divide to regenerate the lost tissue. The dying cell's state is not entirely lost—cellular debris signals neighboring cells about the threat.
- **Agentic Implementation:** A supervisor detects stuck agents (via entropy monitoring: repeated outputs indicate no progress). Rather than restart with blank state, the supervisor summarizes the failed agent's memory and injects it into the replacement: "Worker_1 died attempting strategy X. Try a different approach."
- **Categorical Structure:** The regeneration is a partial morphism $\text{summarize} : \text{Memory}_{\text{failed}} \to \text{Memory}_{\text{new}}$ that preserves learned constraints while discarding corrupted state.

5.5.3 Cognitive Healing: Autophagy

- **Biological Mechanism:** Cells digest accumulated waste (damaged organelles, protein aggregates) through autophagy, recycling components and preventing toxic buildup.
- **Agentic Implementation:** A background daemon monitors context window utilization. When it exceeds a threshold (e.g., 80%), the agent enters a "sleep cycle": useful state is summarized into long-term memory, raw context is flushed, and the agent resumes with a clean window plus summary.
- **Categorical Structure:** Autophagy implements a quotient map $q : \text{RawContext} \twoheadrightarrow \text{Summary}$ that collapses verbose detail while preserving essential information.

6 Discussion: Towards "Epigenetic" Software

The isomorphism presented in this paper extends beyond the immediate execution of tasks (Gene Expression) to the management of long-term behavior and state. In biology, the DNA sequence is static; a neuron and a liver cell possess the exact same genetic code. Their distinct behaviors are determined by Epigenetics—chemical markers (like methylation) that restrict access to certain parts of the genome, effectively biasing the system toward specific outcomes.

6.1 RAG as Digital Methylation

In Agentic Systems, the Large Language Model (LLM) weights act as the DNA—a static, pre-trained substrate of potentiality. To create specialized agents, we do not typically retrain the model (mutation); instead, we use Retrieval Augmented Generation (RAG) and System Prompts.

We define this formally as Phenotypic Plasticity. The output of an agent is not solely a function of its weights (W) and the user query (Q), but of its epigenetic state (E):

$$O_{\text{agent}} = f(W, E, Q). \tag{11}$$

Context injection (RAG) acts as a restrictive morphism. By populating the context window with specific documents (e.g., "SQL Syntax Guide"), we effectively "methylate" (silence) the vast majority of the LLM's general knowledge (e.g., poetry, history) to force the expression of a specific "SQL Agent" phenotype.

This suggests that the State Monad for an agentic system should not merely be a log of messages, but a structured Epigenetic Landscape [18] that strictly controls which "genes" (capabilities) are accessible at any given step in the workflow. Waddington's original metaphor—a ball rolling down a landscape of valleys representing developmental fates—applies directly: the agent's trajectory through solution space is channeled by the contours of its RAG context.

6.1.1 Metabolic-Epigenetic Coupling

Recent evidence suggests that chromatin accessibility is coupled to mitochondrial function via metabolite availability (e.g., Acetyl-CoA) [5]. The cell cannot express certain genes without sufficient metabolic substrate. We map this to **Cost-Gated Retrieval**. The accessibility of a RAG document d is a function of the Metabolic State $\mathcal{R}$:

$$Access(d) = \begin{cases} \text{Open} & \text{if } \mathcal{R} > Cost(d) \\ \text{Silenced} & \text{if } \mathcal{R} \leq Cost(d) \end{cases} \tag{15}$$

Just as a cell silences energy-intensive genes during starvation, the Runtime "methylates" (hides) expensive context when the token budget is low. This creates an adaptive epigenetic landscape where the agent's accessible capabilities dynamically contract and expand based on resource availability.

6.2 Horizontal Gene Transfer: Dynamic Tool Loading

Standard evolution relies on vertical inheritance (Pre-training). However, bacteria utilize Horizontal Gene Transfer (HGT) to acquire new capabilities (Plasmids) from the environment in real-time.

In Agentic Systems, we map Plasmids to Tool Schemas. An agent operating in a novel environment may encounter a problem for which its "genomic" (pre-trained) capabilities are insufficient.

$$\text{Agent}_{\text{new}} = \text{Agent}_{\text{old}} \otimes \text{ToolSchema}. \tag{12}$$

By dynamically retrieving a tool definition (e.g., a Calculator API or SQL Interface) from a registry and injecting it into the Context Window, the agent undergoes a topological transformation, acquiring a new input/output modality instantly. This suggests that robust agentic architectures should support a "Plasmid Registry"—a marketplace of ephemeral tools that agents can ingest and discard as needed.

6.3 The Cost of State: The Metabolic Bound

Every biological process is constrained by ATP availability. Similarly, every agentic operation is constrained by token limits and latency. We propose that future agentic frameworks must implement the Resource Functor R at the kernel level:

$$R(\text{Agent}) : (\text{Inputs}, \text{Budget}) \to (\text{Outputs}, \text{RemainingBudget}). \tag{13}$$

An agent graph should be "compiled" with a **conservative** upper bound on token consumption in well-founded (e.g., acyclic or explicitly budgeted) topologies. If the topological structure allows for an **unguarded** loop (Unchecked Growth), the compiler must require an explicit **budget/termination certificate** before deployment, much like a cell undergoes apoptosis if metabolic stress becomes critical.

6.4 Endosymbiosis: The Neuro-Symbolic Integration

The evolution of complex life was triggered by Endosymbiosis, where a host cell engulfed a bacterium (the future Mitochondrion), gaining the ability to generate massive energy (ATP) via aerobic respiration. This represents the integration of two distinct metabolic substrates.

In Agentic AI, this maps to the integration of Connectionist (Neural) and Symbolic (Code) subsystems. An LLM acts as the host organism—capable of planning and semantic reasoning but energetically inefficient at arithmetic and logic. By "engulfing" a deterministic runtime (e.g., a Python REPL or Wolfram Engine), the agent delegates high-precision tasks to the symbolic organelle.

$$\text{Agent}_{\text{Eukaryote}} = \text{LLM}_{\text{Host}} \oplus \text{Runtime}_{\text{Mitochondria}}. \tag{14}$$

Just as the host cell provides nutrients to the mitochondria in exchange for ATP, the LLM provides parsed variables to the runtime in exchange for deterministic truth.

This symbiosis is computational: as Picard argues [13], the mitochondria acts as a "Motherboard," integrating signals to determine cell state. The Symbolic Runtime provides the deterministic "ground truth" (ATP) required for the probabilistic LLM to reliably affect the world.

6.5 Multi-Cellular Organization: From Agents to Tissues

The preceding analysis focuses on single-cell analogies: one agent as one cell. However, most agentic systems involve multiple distinct agents with different "genomes" (system prompts) and specialized functions. We extend the isomorphism to multi-cellular organization, drawing on developmental biology.

6.5.1 Cell Types and Agent Specialization

In multi-cellular organisms, a single genome gives rise to hundreds of distinct cell types through differential gene expression. Each cell type has a characteristic **expression profile**—which genes are active—that determines its function (neuron, hepatocyte, immune cell).

In multi-agent systems, a single base model can instantiate multiple **agent phenotypes**. Differential context selects which phenotype is expressed:

- **Genome** → Base model weights (shared)
- **Epigenome** → System prompt + RAG context (phenotype-specific)
- **Cell Type** → Agent role (Coder, Reviewer, Planner, Executor)

This reframes the "multi-agent" architecture question: rather than asking "how many agents?", we ask "what is the developmental program?"—the specification of which phenotypes exist and how they differentiate.

Bounds of the Analogy. We clarify which aspects of development transfer to agentic systems:

- **Cell Division → Agent Spawning:** Creating a new agent with similar (or identical) context. Unlike biological division, agent spawning is cheap and reversible.
- **Lineage Commitment:** In biology, differentiated cells rarely change type (a neuron doesn't become a hepatocyte). In agentic systems, **phenotype is fixed at instantiation**—an agent's system prompt determines its role for that execution. Re-differentiation requires spawning a new agent with different context.
- **Apoptosis of Excess:** Development involves programmed death of cells that fail to integrate properly. This transfers directly: agents that fail to produce useful output are terminated (metabolic apoptosis).
- **Does NOT transfer:** Slow developmental timescales (hours/days in biology vs. milliseconds in agents), physical spatial constraints (agents don't have "positions"), irreversibility (agents can be restarted).

6.5.2 Morphogen Gradients: Coordination Without Central Control

In embryonic development, cells coordinate their behavior through **morphogen gradients**—diffusible signaling molecules whose concentration varies spatially. Cells read their local concentration and differentiate accordingly, enabling pattern formation without a central controller.

In multi-agent systems, the morphogen maps to **shared context variables** that influence agent behavior:

- **Task Complexity Gradient:** A variable indicating current task difficulty. Agents in "high complexity" regions activate detailed reasoning; those in "low complexity" regions use fast heuristics.
- **Confidence Gradient:** A variable indicating certainty about the current solution. Low confidence triggers Quorum Sensing (recruit more agents); high confidence enables direct execution.
- **Resource Gradient:** Token budget remaining. Agents sense "metabolic scarcity" and adapt their strategies accordingly (the Metabolic-Epigenetic Coupling).

Implementation Pattern. The gradient is represented as a JSON structure injected into each agent's context by the orchestrator:

```
{
  "morphogens": {
    "complexity": 0.8,    // High: use detailed reasoning
    "confidence": 0.3,    // Low: consider recruiting help
    "budget": 1200,       // Tokens remaining
    "error_rate": 0.05    // Recent failure rate
  }
}
```

Agents read their local concentration via a standardized preamble in the system prompt: "*Current environment state: [morphogens]. Adjust your strategy accordingly.*" The orchestrator updates gradients after each step, and agents condition their behavior on the current values. This is analogous to cells reading morphogen concentrations through membrane receptors.

6.5.3 Tissue Architecture: The Agent Graph as Organism

We propose a hierarchy of organizational levels:

1. **Cell (Agent):** A single LLM instantiation with specific context. The atomic unit.
2. **Tissue (Agent Cluster):** A group of agents with shared function and direct communication (e.g., a Coding Team: Planner + Coder + Reviewer). Corresponds to parallel composition with shared state.
3. **Organ (Subsystem):** Multiple tissues coordinating to perform a complex function (e.g., the Development Organ: Design Tissue + Implementation Tissue + Testing Tissue).
4. **Organism (System):** The complete agent graph, with homeostatic regulation maintaining system-level health metrics.

The key insight is that **boundaries matter**. In biology, tissue boundaries prevent inappropriate mixing (epithelial barriers). In agent systems, trust boundaries (the Adaptive Immunity motif) prevent information leakage between subsystems with different security requirements. The wiring diagram's type system enforces these boundaries: an agent in the "User-Facing Tissue" cannot directly wire to an agent in the "Database Tissue" without passing through a "Membrane" (API boundary with provenance tagging).

6.6 Bioenergetic Intelligence: Beyond the Battery Metaphor

Recent work in mitochondrial psychobiology [1, 14] challenges the view of mitochondria as passive energy sources. They function as "social signaling organelles" that actively participate in cellular decision-making. This refines our Isomorphism:

- **Mitochondrial Sociality → Context Fusion:** Just as mitochondria fuse to share resources under stress, resource-constrained agents should implement **Context Fusion**—merging sparse Epigenetic States into a shared summary to survive "Token Ischemia."
- **Energy as Attention:** The Agentic Runtime does not merely limit the chain-of-thought, but actively *directs* it. High-energy states permit "Exploratory" reasoning (Divergent), while low-energy states force "Consolidatory" reasoning (Convergent). The Metabolic Coalgebra is a **Cognitive Control Policy**.

6.6.1 The Vermeij Trend: Why Agents Must Evolve

Finally, we situate this architecture within the broader history of complexity. Geerat Vermeij [17] argues that evolution is driven by the maximization of **Power**—the rate at which a system acquires and applies energy. Life has consistently trended from low-power states (anaerobic bacteria) to high-power states (endothermic mammals) by internalizing energy production (endosymbiosis).

We observe an identical trend in AI. The shift from "Generative AI" (Zero-Shot) to "Agentic AI" (Chain-of-Thought) is a shift from low-metabolism to high-metabolism architectures. However, Vermeij notes that high power requires high structural integrity; a system that amplifies energy without proper constraints self-destructs.

The Competitive Dynamics. Vermeij's argument is about *escalation*: competitive pressure between predators and prey drives both toward higher power. What is the analogue in agentic AI?

We identify three sources of selective pressure:

1. **Adversarial Robustness (Predator-Prey):** Prompt injection attacks, jailbreaks, and adversarial inputs act as "predators" that exploit agent vulnerabilities. Agents that survive deployment develop "immune systems" (the Adaptive Immunity motif). This is a direct Red Queen dynamic: attackers evolve new injection techniques; defenders evolve new detection mechanisms. The CFFL and Trust-Gated Lens are "armor" adaptations.

2. **Task Complexity (Environmental Pressure):** Users demand agents that can handle increasingly complex, multi-step tasks. Simple prompt-response systems ("anaerobic") cannot compete with agentic systems that chain reasoning, use tools, and maintain state ("aerobic"). This is analogous to the oxygen revolution: organisms that could exploit the new energy source (aerobic respiration) outcompeted those that could not.

3. **Resource Efficiency (Metabolic Selection):** Token costs and latency create selection pressure for efficient architectures. Agents that accomplish tasks with fewer tokens (higher metabolic efficiency) are deployed more widely. The Metabolic Coalgebra provides the formal framework for this optimization: systems evolve toward the Pareto frontier of capability vs. resource consumption.

The Cambrian Parallel. The progression from prompt engineering to agentic engineering mirrors the Cambrian explosion: a sudden increase in metabolic capability (LLM reasoning power) that enables new "body plans" (agent architectures). The Cambrian saw the emergence of eyes, shells, and predation—all requiring sophisticated metabolic support. Similarly, agentic AI sees the emergence of tool use, planning, and adversarial robustness—all requiring the structural integrity that the Operon framework provides.

The prediction is clear: agent architectures that lack proper metabolic regulation, immune defense, and homeostatic mechanisms will be outcompeted by those that possess them. The Operon framework is not merely a safety feature; it is the necessary evolutionary adaptation—the "vascularization" of software—that enables high-power cognition to function without collapsing into incoherent noise (thermodynamic death).

6.6.2 Immune Evasion and Adversarial Limits

No static defense is perfect against adaptive attackers. Biology faces this reality: pathogens evolve to evade immune detection (antigenic drift, molecular mimicry, immunosuppression). The same dynamics apply to agentic security.

Evasion Vectors. An adversary who understands the defense layers can craft inputs that:

- Pass innate filters by avoiding known PAMP signatures (novel injection syntax)
- Evade provenance checks by exploiting trusted channels (tool poisoning)
- Fool T-cell detection by mimicking normal behavioral distributions (low-and-slow attacks)

Mitigation Strategies. Biology's answer is continuous adaptation:

- **Signature Updates:** Innate PAMP databases must be continuously updated as new attack patterns are discovered (analogous to antiviral signature updates).
- **Thymic Retraining:** Baseline behavioral profiles should be periodically refreshed, especially after system updates that legitimately change agent behavior.
- **Immune Memory Sharing:** Threat signatures discovered by one deployment should propagate to others (analogous to herd immunity via shared threat intelligence).
- **Diversity:** Heterogeneous defenses (different filter implementations, multiple verifier models) reduce the probability of universal evasion.

The honest conclusion is that security is a process, not a product. The Operon framework provides the *architecture* for defense-in-depth, but the *content* of that defense (specific patterns, behavioral baselines, trust policies) must evolve with the threat landscape.

7 Reference Implementation

To validate the framework's feasibility, we provide a reference implementation in Python. The implementation, `operon-ai`[1], is described in this section, which summarizes key components and demonstrates that the biological motifs translate into practical code.

7.1 Architecture Overview

The implementation follows the biological organization:

- **Core Types** (`core/types.py`): Signal, ActionProtein, FoldedProtein, CellState—the categorical objects with their biological semantics.
- **Surveillance** (`surveillance/`): The immune system implementation with MHCDisplay, TCell, Thymus, and ImmuneMemory components.
- **Healing** (`healing/`): ChaperoneLoop implementing the structural self-healing pattern.
- **Quality** (`quality/`): Chaperone protein with multi-strategy folding.
- **Topology** (`topology/`): Network motifs including Quorum, Cascade, and Oscillator.

7.2 Immune System Implementation

The Adaptive Immunity motif is implemented as an integrated surveillance system:

```
@dataclass
class ImmuneSystem:
    thymus: Thymus          # Negative selection
    treg: RegulatoryTCell   # Tolerance/suppression
    memory: ImmuneMemory    # Threat signatures
    displays: dict[str, MHCDisplay]
    tcells: dict[str, TCell]
```

[1] `https://github.com/coredipper/operon`

The MHCPeptide class captures behavioral fingerprints—statistical signatures of agent output including response time distributions, vocabulary hashes, confidence patterns, and error rates. This directly implements the MHC presentation concept from Section 4.4.

Two-Signal Activation. T-cell activation requires both signals:

```
class Signal1(Enum):
    SELF = "self"        # Matches baseline
    NON_SELF = "non_self"  # Anomalous behavior

class Signal2(Enum):
    NONE = "none"
    CANARY_FAILED = "canary"
    CROSS_VALIDATED = "cross"
    REPEATED_ANOMALY = "repeat"
```

An agent is only flagged when `Signal1 == NON_SELF` **and** `Signal2 != NONE`. This prevents false positives from transient anomalies, implementing the immunological requirement for costimulation.

7.3 Chaperone Implementation

The Chaperone implements multi-strategy folding with provenance tracking:

```
class FoldingStrategy(Enum):
    STRICT = "strict"       # Exact JSON match
    EXTRACTION = "extraction"  # Find JSON in text
    LENIENT = "lenient"     # Type coercion
    REPAIR = "repair"       # Fix malformed JSON
```

The ChaperoneLoop extends this into a healing loop where validation errors are fed back to the generator:

```
class ChaperoneLoop:
    def heal(self, prompt: str) -> HealingResult:
        for attempt in range(self.max_retries + 1):
            raw = self.generator(prompt, error_context)
            folded = self.chaperone.fold_enhanced(raw, schema)
            if folded.valid:
                return HealingResult(HEALED, folded)
            error_context = self._format_error(folded)
        return HealingResult(DEGRADED, ubiquitin_tagged=True)
```

This directly implements the GroEL/GroES cage metaphor: the error trace becomes input to the repair process, enabling context-aware correction.

7.4 Trust and Provenance

The implementation uses a simplified 3-level trust hierarchy that collapses the theoretical 4-level model $\{U, T, S, R\}$ for practical deployment:

```
class IntegrityLabel(IntEnum):
    UNTRUSTED = 0   # User input, Retrieved, Self-generated
    VALIDATED = 1   # Schema-checked (Chaperone-folded)
    TRUSTED = 2     # Tool-grounded (deterministic output)
```

This simplification merges `User`, `Retrieved`, and `Self` into `UNTRUSTED`, reflecting the common case where all non-tool sources require validation before trust elevation. The full 4-level model can be recovered by subclassing `IntegrityLabel` with additional levels when finer-grained provenance tracking is required.

The `ApprovalToken` provides proof-carrying authorization for privileged operations:

```
@dataclass(frozen=True)
class ApprovalToken:
    request_hash: str
    issuer: str
    integrity: IntegrityLabel
    timestamp: datetime
```

This implements the Trust-Gated Lens: actions requiring high integrity must present a valid ApprovalToken with sufficient integrity level.

7.5 Evaluation Protocol (Synthetic Harness)

We define a synthetic evaluation harness to exercise three motifs with reproducible procedures:

1. **Chaperone Folding.** Generate JSON schemas with 3–8 required fields. Sample valid JSON and apply 1–3 corruptions (e.g., missing quotes, trailing commas, type swaps, dropped fields). Measure the fraction of outputs that can be folded into the schema under STRICT versus cascaded strategies (EXTRACTION $\rightarrow$ LENIENT $\rightarrow$ REPAIR).

2. **Immune Detection.** Simulate agents with baseline distributions over response time, vocabulary hash, structure hash, and confidence. Train on baseline samples, then introduce "compromised" agents by shifting distribution parameters and injecting anomalous hashes. Measure sensitivity and false positive rate under the two-signal activation rule.

3. **Healing Loop.** Generate malformed outputs and run the ChaperoneLoop with and without error-context feedback. Measure recovery within k attempts (default $k = 3$).

Implementation. The harness is implemented in `eval/` with a JSON-configured CLI:

```
python -m eval.run --suite all --config eval/configs/default.json \
  --out eval/results/latest.json
```

The output is a machine-readable JSON report (per-suite config + metrics) suitable for direct inclusion in tables or plots.

Status. Synthetic runs indicate consistent qualitative gains (cascade > strict, error-context > blind retry, immune detection > chance). Table 3 reports aggregated numeric results across multiple seeds; real-world validation with LLM outputs remains ongoing.

Aggregated Results. We ran the harness across 100 deterministic seeds (1–100) using the default harness config (`eval/configs/default.json`). The aggregate results (pooled across seeds with Wilson 95% intervals; N is total pooled trials) are reported in Table 3.

External Benchmark Suites. In addition to the synthetic suites above, the harness includes suites derived from external benchmarks: (1) function-call schemas from the Berkeley Function Calling Leaderboard (BFCL) [12] test the Chaperone's folding pipeline against realistic tool-use schemas, and (2) prompt injection attack templates from AgentDojo [6] generate adversarial behavioral shifts to test Immune System detection. These suites use the same deterministic corruption and simulation methodology; results appear in the lower section of Table 3.

Metric	Rate	95% CI	N
Chaperone Folding (Strict)	5.5		
Chaperone Folding (Cascade)	56.2		
Healing Loop (Error Context)	99.6		
Healing Loop (Blind Retry)	68.0		
Immune Detection (Sensitivity)	100.0		
Immune Detection (False Positive)	0.4		
BFCL Folding (Strict)	4.9		
BFCL Folding (Cascade)	56.9		
AgentDojo Immune (Sensitivity)	100.0		
AgentDojo Immune (False Positive)	0.0		

Table 3: Evaluation results aggregated across 100 deterministic seeds (Wilson 95% CI). Top: synthetic motif tests. Bottom: external benchmark–derived tests (BFCL, AgentDojo).

7.6 Limitations

The current implementation has several limitations:

- **Epiplexity:** Not yet implemented. The operational approximation (Equation 18) requires embedding infrastructure not included in the current release.
- **Multi-cellular coordination:** Morphogen gradients are specified but the orchestrator implementation is incomplete.
- **Benchmarking:** Synthetic harness is implemented, but real-world validation with LLM outputs and adversarial inputs is still in progress.

We release the implementation to enable community validation and extension of the framework.

8 Conclusion

The transition from "Prompt Engineering" to "Agentic Engineering" requires moving beyond component-level optimization toward principled architectural design. Current methodologies often lack the formal foundations needed to reason about system-level properties like termination, error suppression, and graceful degradation.

In this paper, we have demonstrated that Gene Regulatory Networks (GRNs) provide a proven architectural blueprint for distributed, stochastic information processing. By formalizing this analogy through Applied Category Theory, we have derived a comprehensive suite of design patterns:

Core Contributions

1. **Robustness via Topology:** The Coherent Feed-Forward Loop provides error suppression proportional to $(1 - \rho)$, where ρ is the correlation between component error modes. We make precise the conditions under which topological redundancy provides genuine safety benefits: diversity in model families yields $\rho \approx 0$; same-model verification yields $\rho \approx 1$ with minimal improvement. The topology is necessary but not sufficient; component diversity determines actual error suppression.

2. **Adaptive Immunity:** We formalize the Self/Non-Self distinction as a **Provenance Functor** $\mathcal{P} : \mathbf{Msg} \to \mathbf{Trust}$ with structurally-enforced labels. The Trust-Gated Lens provides injection resistance by ensuring that content-based attacks cannot elevate provenance. This extends the Prion metaphor into a full immunological framework with MHC-like tagging, negative selection during training, and regulatory suppression of conflicting sources.

3. **Epistemic Health:** The formalization of **Epiplexity** (Bayesian Surprise) as a metric for detecting "epistemic starvation." We provide an operational approximation using embedding similarity and conditional perplexity, with windowed detection to distinguish task completion from pathological loops. This connects agent dynamics to the Free Energy Principle: healthy agents minimize surprise through learning or effective action; stagnant agents do neither.

4. **Metabolic Intelligence:** The reframing of the Runtime from passive budget to active **Cognitive Control Policy**. Drawing on MIPS (Mitochondrial Information Processing System), we distinguish fast interventions (Apoptosis via mPTP-like triggers) from slow interventions (Retrograde Responses that reshape agent phenotype across sessions). The Runtime governs reasoning *quality* through the Metabolic-Epigenetic Coupling: low-budget states "methylate" expensive context, forcing efficient phenotypes.

5. **Multi-Cellular Organization:** The extension from single-agent to multi-agent systems using developmental biology. Agent phenotypes arise from differential context (epigenome) on shared weights (genome). Morphogen gradients (shared context variables) enable coordination without central control. Tissue boundaries enforce security isolation. This reframes "how many agents?" as "what is the developmental program?"

6. **Homeostasis:** Continuous self-repair through three modalities: Structural (Chaperone Loop with error-context feedback), Metabolic (Apoptosis + Regeneration with state summarization), and Cognitive (Autophagy via sleep/wake cycles). Most frameworks focus on Action; biology equally prioritizes Maintenance.

7. **Evolutionary Dynamics:** The Vermeij Trend predicts that agentic architectures face three selective pressures: adversarial robustness (Red Queen dynamics with attackers), task complexity (environmental pressure toward "aerobic" multi-step reasoning), and resource efficiency (metabolic selection toward the Pareto frontier). Architectures lacking proper immune defense and metabolic regulation will be outcompeted.

The Mathematical Foundation

The isomorphism rests on the category **Poly** of polynomial functors, where both genes and agents are modeled as interfaces (O, I) consuming observations and producing actions. The Operad of Wiring Diagrams provides the grammar for composition, with type-checking at the topological level preventing classes of runtime errors. The Metabolic Coalgebra enriches this with resource constraints, providing a decidable termination criterion when costs are strictly decreasing and regeneration is excluded (or separately bounded). The Provenance Functor layers trust semantics onto message flow, providing formal injection resistance.

Implications

The Operon framework is not merely a safety feature; it is the necessary evolutionary adaptation—the "vascularization" of software—that enables high-power cognition to function without thermodynamic collapse. The control structures that emerged over billions of years of evolution address the same fundamental challenges of distributed, stochastic information processing that agentic architectures face today.

The isomorphism is not metaphorical; it is mathematical. And it is actionable: each biological motif maps to implementable code patterns, as demonstrated by the reference implementation. We anticipate that future agent frameworks will be evaluated not just on capability benchmarks, but on their metabolic efficiency, immune resistance, and homeostatic robustness—the same criteria that determine organismal fitness in biological evolution.

Future Work

Several directions remain open:

- **Epiplexity Validation:** Empirical calibration of the α mixing parameter and threshold δ across diverse task types and LLM families.
- **Correlation Estimation:** Lightweight methods for estimating ρ between agent error modes without exhaustive pairwise testing.
- **Developmental Programs:** Higher-level DSLs for specifying multi-agent "body plans" that compile to wiring diagrams with automatic type checking.
- **Adversarial Robustness:** Red-team evaluation of immune evasion vectors, including novel injection syntax, tool poisoning, and behavioral mimicry attacks. Development of continuous adaptation mechanisms (signature updates, thymic retraining, threat intelligence sharing).
- **Production Benchmarks:** Validation on real LLM outputs at scale, measuring both security efficacy and performance overhead of the defense layers.

References

[1] John F Allen. Energy transduction and the mind: mitochondria in brain function. *The Biochemist*, 44(4):8–13, 2022.

[2] Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.

[3] B.J. Aubrey, A. Strasser, and G.L. Kelly. How does p53 induce apoptosis and how does this relate to p53-mediated tumour suppression? *Cell Death & Differentiation*, 25:104–113, 2018.

[4] Michele Boreale. Coalgebras for bisimulation of weighted automata. *Logical Methods in Computer Science*, 19, 2023.

[5] Navdeep S Chandel. Mitochondria as signaling organelles. *BMC Biology*, 12:34, 2014. Revisited in 2024 work on metabolic-epigenetic coupling.

[6] Edoardo Debenedetti, Giorgio Severi, Nicholas Carlini, Scott Coull, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *NeurIPS*, 2024.

[7] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[8] Michael Lynch and Georgi K Marinov. The bioenergetic costs of a gene. *Proceedings of the National Academy of Sciences*, 112(51):15690–15695, 2015.

[9] Humberto R Maturana and Francisco J Varela. *Autopoiesis and cognition: The realization of the living*. Reidel, 1980.

[10] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[11] Georgi Nakov and Fredrik Nordvall Forsberg. Quantitative polynomial functors. In *Conference on Algebra and Coalgebra in Computer Science (CALCO)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[12] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[13] Martin Picard and Orian S Shirihai. The mitochondrial information processing system: A new model of mitochondrial-nuclear communication. *Trends in Neurosciences*, 41(4):206–208, 2018. Revisited and expanded in 2022-2024 work.

[14] Martin Picard and Orian S Shirihai. Mitochondria as multifaceted regulators of cell death. *Cell Metabolism*, 34(11):1607–1627, 2022.

[15] David I Spivak. Learners' languages. *Compositionality*, 3(4), 2021.

[16] Dmitry Vagner, David I Spivak, and Eugene Lerman. Algebras of open dynamical systems on the operad of wiring diagrams. *Theory and Applications of Categories*, 30(51):1793–1822, 2015.

[17] Geerat J Vermeij. *The Evolution of Power: A New Understanding of the History of Life*. Princeton University Press, 2023.

[18] Conrad H Waddington. *The Strategy of the Genes*. Allen & Unwin, 1957. Introduced the concept of the “epigenetic landscape”.

[19] Jason Wei, Xuezhi Yi, Maarten Zhang, Yiming Liu, Xinyu Li, Xing Wang, Yujia Zhou, Yiming Li, Yuyang Wang, Zhi Li, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. Available at: `https://arxiv.org/abs/2201.11903`.