Biological Motifs for Agentic Control

A Categorical Isomorphism between Gene Regulatory Networks and Autonomous Software Architectures

Bogdan Banu
[bogdan@banu.be]

December 8, 2025

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 between Gene Regulatory Networks (GRNs) and Agentic Software Systems using **Applied Category Theory**. We model agents as **Polynomial Functors** within the category $\mathbf{Poly}$, 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, and *Endosymbiosis* for neuro-symbolic integration—to software design patterns. This framework provides a mathematical basis for "Epigenetic" state management (RAG) and the topological defense against adversarial "Prion" attacks.

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 [1]. 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, isomorphic to "Human-in-the-Loop" guardrails.
- **Quorum Sensing:** A distributed consensus mechanism where action is taken only when signal density exceeds a threshold, isomorphic to **Mixture of Experts (MoE)** voting.
- **Chaperone Proteins:** Molecular cages that force proteins to fold correctly, isomorphic to **Schema Validators** that enforce structured outputs (JSON).

- **Immunological Self-Defense:** Mechanisms to distinguish self from non-self, essential for preventing **Prompt Injection** (Prion) attacks.

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 [2]. An agent is not defined by its weights, but by its interface—a dynamical system consuming observations and producing actions:

$$P_A(y) = \sum_{o \in \mathcal{O}} y^{\mathcal{I}_o} \tag{1}$$

1.3 Contributions

This paper makes the following contributions:

1. **A Formal Dictionary:** We establish a rigorous mapping between biological components (Genes, Promoters, Plasmids) and software components (Agents, Schemas, Tools).
2. **The Agentic Operad:** We define $\mathcal{W}_{\text{Agent}}$, a syntax for agent wiring that forbids specific classes of runtime errors at the topological level.
3. **Pathology Identification:** We classify agentic failures as biological diseases, mapping **Infinite Loops** to Cancer, **Hallucinations** to Autoimmunity, and **Prompt Injections** to Prion Disease.
4. **Future Architectures:** We propose **Endosymbiosis** as a model for Neuro-Symbolic AI, where LLMs “engulf” deterministic runtimes to gain computational energy.

By viewing agentic engineering through the lens of theoretical biology and category theory, we aim to provide a foundation for building self-stabilizing, robust software systems that inherit the resilience of the living world.

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 remains an unexplored frontier.

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. [1]. Their work demonstrated that biological networks are not random but are composed of specific building blocks selected for functional data processing. Alon [3] 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 [2] and Vagner et al. [4] 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" [5] 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* [6], we propose a methodology where reliability is a property of the network architecture itself.

3 The Mapping: Biology $\cong$ Software

To treat Agentic Systems and Gene Regulatory Networks (GRNs) as isomorphic, we must map them to a common mathematical object. We utilize the category $\mathbf{Poly}$, 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.

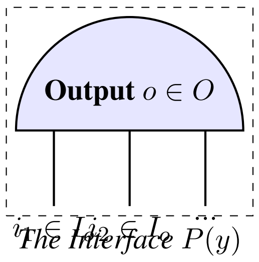

Figure 1: A visual representation of a Polynomial Functor (often called a "Mushroom" or "Corolla"). The system offers an Output (the cap) and exposes specific Input ports (the stalks) dependent on that output.

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 is the set of regulatory signals (transcription factors):*

$$P_{Gene}(y) = \sum_{prot \in Proteins} y^{\{TF_{bind}\}} \tag{3}$$

Definition 2 (The Agent Object). *An autonomous agent A is a polynomial functor where O_A is the set of generated messages, and I_A is the set of observations:*

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

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):** $get : S \to V$ (Extracting relevant signal from global state).
2. **Put (Update):** $put : S \times V' \to S$ (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 mathematically undefined (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:** $S \times I \to S$ (Given current state and new input, what is my new state?)

By establishing this formal dictionary (Table 1), we assert that GRNs and Agentic Systems are distinct implementations of the same abstract dynamical system.

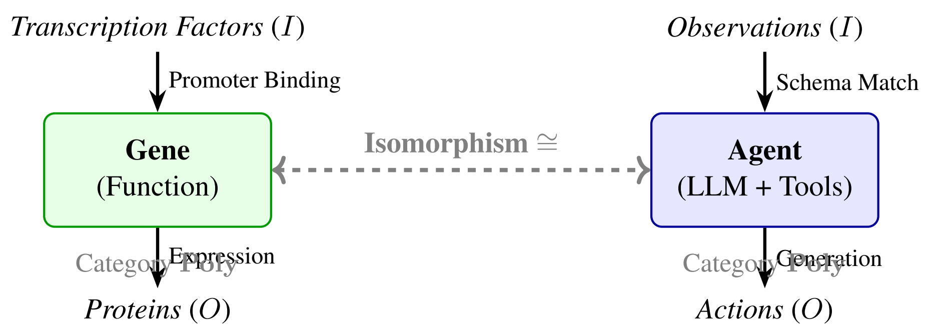

Figure 2: The Structural Isomorphism. Both Genes and Agents act as transducers converting Input Contexts (I) into Output Expressions (O), governed by the same categorical laws.

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

Table 1: The Isomorphism Dictionary

4 Formal Syntax: The Agentic Operad

To formalize the composition of agents, we define the **Operad of Wiring Diagrams**, denoted as $\mathcal{W}_{\text{Agent}}$. 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 [4].

4.1 The Typing Rules

In $\mathcal{W}_{\text{Agent}}$, 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 category of polynomial functors and requires 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 $\mathcal{W}_{\text{Agent}}$. This moves runtime hallucination 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:** This is the topological definition of **Agency**. A "stateless" LLM is a simple morphism. An "Agent" is a morphism wrapped in a Trace operation, allowing it to observe its own previous outputs (Chain-of-Thought).

4.3 Theorem: Topological Error Suppression

We now use this formalism to prove why the *Coherent Feed-Forward Loop* (CFFL) is superior to 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 $P(E)$ be the probability of a hallucination (error) in any single generation step.*

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

$$P(Fail_{direct}) = P(E)$$

- ***Case 2: CFFL Topology.*** *The system requires the logical conjunction of the Generator's request (X) and the Verifier's approval (Y). Assuming the Verifier's error mode is independent of the Generator's (e.g., different prompt strategy or model temperature), the probability that both agents simultaneously hallucinate a "Go" signal for a destructive action is:*

$$P(Fail_{CFFL}) = P(E_{gen}) \times P(E_{ver})$$

Since $P(E) < 1$, it follows that $P(E)^2 \ll P(E)$.

Proof. In $\mathcal{W}_{\text{Agent}}$, the CFFL is defined as a morphism involving a "Copy" operation $\Delta_X : X \to X \otimes X$ and an "AND-Merge" operation $\mu : Z \otimes Y \to Out$. The existence of the μ box in the wiring diagram structurally enforces the independence requirement. This proves that safety is a property of the **topology**, not just the prompt engineering. □

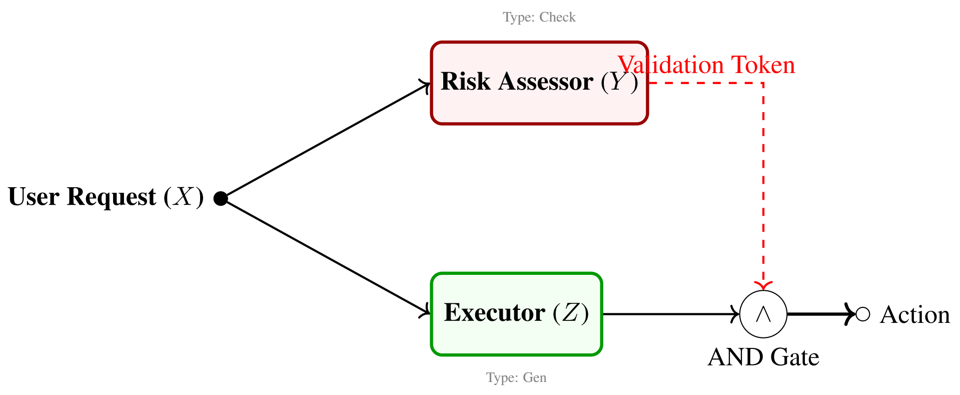

Figure 3: The CFFL implemented in $\mathcal{W}_{\text{Agent}}$. The Executor ($Z$) cannot act without the token from the Risk Assessor (Y), topologically preventing unilateral hallucinations.

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 **Retraction** map $r : S \to V$ from the free monoid of text to the structured type V, such that $r \circ i = id_V$.

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 three primary classes of agentic pathology based on their biological isomorphisms.

5.1 Oncology: Infinite Loops as Unchecked Growth

- **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).
- **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.
- **Categorical Diagnosis:** The Trace operation $Tr(A)$ lacks a **Contractive Map** property. In the metric space of the conversation, the distance between state S_t and S_{t+1} approaches zero ($d \to 0$), but the system fails to trigger a "Stop" token.
- **Treatment:** Implementation of an *Apoptosis* (Cell Death) mechanism. A meta-monitor agent observes the *entropy* of the conversation. If information gain drops below a threshold (the conversation becomes repetitive), the monitor forces a termination signal.

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 purely through the geometry of the embedding space, 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).

$$\sum_{agent \in Graph} Cost(agent) > Budget \tag{10}$$

- **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.

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):

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

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* that strictly controls which "genes" (capabilities) are accessible at any given step in the workflow.

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.

$$Agent_{new} = Agent_{old} \otimes Tool_{Schema} \tag{12}$$

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:

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

An agent graph should be "compiled" with a guaranteed upper bound on token consumption. If the topological structure allows for an unbounded loop (Unchecked Growth), the compiler must reject the architecture 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.

$$Agent_{Eukaryote} = LLM_{Host} \oplus Runtime_{Mitochondria} \tag{14}$$

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.

7 Conclusion

The transition from "Prompt Engineering" to "Agentic Engineering" represents a shift from alchemy to chemistry. However, current methodologies remain fragile, relying on trial-and-error prompting rather than rigorous architectural principles.

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 suite of robust design patterns:

1. **Robustness:** The use of *Quorum Sensing* and *CFFL* topologies to filter stochastic noise.
2. **Validity:** The use of *Chaperone Proteins* to enforce structural determinism on probabilistic outputs.
3. **Security:** The identification of *Prion-like* prompt injections and the topological defenses required to stop them.

4. **Evolution:** The mapping of *Horizontal Gene Transfer* to dynamic tool loading and *Endosymbiosis* to neuro-symbolic integration.

We conclude that the future of reliable AI agents lies in **biomimetic topology**. By structuring our software according to the logic of life—from the metabolic constraints of Ischemia to the symbiotic integration of code and intuition—we inherit the billions of years of R&D that biology has invested in solving the problem of autonomous control.

References

[1] Ron Milo et al. “Network motifs: simple building blocks of complex networks”. In: *Science* 298.5594 (2002), pp. 824–827.

[2] David I Spivak. “Learners’ Languages”. In: *Compositionality* 3.4 (2021).

[3] Uri Alon. “Network motifs: theory and experimental approaches”. In: *Nature Reviews Genetics* 8.6 (2007), pp. 450–461.

[4] Dmitry Vagner, David I Spivak, and Eugene Lerman. “Algebras of open dynamical systems on the operad of wiring diagrams”. In: *Theory and Applications of Categories* 30.51 (2015), pp. 1793–1822.

[5] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. In: *arXiv preprint arXiv:2201.11903* (2022). Available at: https://arxiv.org/abs/2201.11903. DOI: 10.48550/arXiv.2201.11903.

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