Biological Motifs for Agentic Control

A Functorial Bridge between Gene Regulatory Networks and Autonomous Software Architectures

Bogdan Banu
bogdan@banu.be

December 9, 2025

Abstract

The transition of Large Language Models (LLMs) from passive generators to autonomous agents has introduced recurring failures in reliability, security, and state management. Current agentic architectures are often constructed ad-hoc, leaving systems vulnerable to cascading errors, runaway feedback loops, and prompt injection attacks.

This paper argues that many of these failure modes are instances of general control problems long addressed by biological networks. We formalize a *functorial correspondence*—at the level of *open-system interfaces and composition*—between Gene Regulatory Networks (GRNs) and Agentic Software Systems using Applied Category Theory. Both are represented as coalgebras of polynomial functors in **Poly**, composed via the operad of wiring diagrams.

We introduce WAgent, a typed syntax for agent wiring that (i) prevents structural integration errors at design time, (ii) enforces explicit validation boundaries through effectful optics and partial validators, and (iii) supports security reasoning via integrity labels and capability/effect tracking. Biological motifs—including coherent feed-forward loops (persistence detection), quorum sensing (consensus), and chaperones (structural validation)—become reusable architectural patterns with precise interface contracts.

We also revisit a common safety claim: wiring alone does not guarantee statistical independence of errors. Instead, topology provides *interlocks* (e.g., two-key execution) while independence is an external design requirement achieved by diversity (models, prompts, tools, and verifiers).

This framework provides a compositional basis for "epigenetic" state management (RAG and memory), for defense against adversarial injection, and for principled termination and resource control in autonomous software.

1 Introduction

Artificial Intelligence is shifting from *Generative AI* (single-shot text generation) to *Agentic AI* (systems that execute multi-step workflows to achieve goals). While individual LLM capabilities scale predictably, engineering *systems of agents* remains fragile. Developers face non-deterministic outputs, runaway loops, adversarial instructions, and the challenge of maintaining global coherence under partial observability and stochastic behavior.

We argue these challenges are not unique to software. They are fundamental constraints of distributed information processing. A close analogue to a multi-agent architecture is a *Gene Regulatory Network* (GRN): thousands of genes read local signals and express proteins that regulate other genes, producing robust behavior under noise, resource constraints, and adversarial environments.

1.1 The Biological Heuristic

Systems biology identifies recurring *network motifs* that implement control functions (filtering, memory, consensus, and defense). We focus on four motifs that map naturally to agentic engineering:

- **Coherent Feed-Forward Loop (CFFL):** persistence detection and gating, analogous to two-key execution guardrails.
- **Quorum sensing:** distributed consensus via thresholded aggregation, analogous to ensemble voting with confidence thresholds.
- **Chaperone-assisted folding:** structural validation and repair loops, analogous to schema validation and retry mechanisms.
- **Immune-like self-defense:** mechanisms distinguishing trusted signals from untrusted inputs, analogous to prompt-injection containment and information-flow policies.

1.2 From Metaphor to Discipline

To make the analogy actionable, we use Applied Category Theory: polynomial functors in **Poly** represent *typed open interfaces*, and wiring diagrams represent *compositional structure*. We emphasize an important precision:

> We do *not* claim GRNs and agentic systems are identical in physics or implementation. We claim they admit a shared *interface-level semantics* in **Poly** that supports compositional reasoning about wiring, validation, gating, and resource control.

1.3 Contributions

1. **A Formal Dictionary:** a rigorous correspondence between biological components (genes, promoters, plasmids) and software components (agents, schemas, tools).
2. **A Safer Agentic Operad** WAgent**:** a syntax for agent wiring with types, integrity labels, and effect tracking, supporting compile-time rejection of ill-formed compositions.
3. **Correct Topological Safety Claims:** topology enforces *interlocks*, not independence; we provide the correct probabilistic statement for CFFL safety and the corresponding design requirements for independence/diversity.
4. **A Pathology Lens:** agentic failures are analyzed as dysregulated network dynamics, informing termination, validation, and security treatments.

2 Related Work

This work sits at the intersection of Systems Biology, Applied Category Theory, and Agentic AI.

2.1 Network Motifs in Systems Biology

Network motifs as statistically over-represented subgraphs were introduced by Milo et al. Their functional characterization, including feed-forward persistence detection, is developed further by Alon.

2.2 Applied Category Theory (ACT)

We use the categorical language of open dynamical systems, polynomial functors, and wiring-diagram operads (e.g., Spivak; Vagner–Spivak–Lerman) to formalize interface composition.

2.3 Reliability in Agentic AI

Techniques such as iterative self-critique and reasoning loops improve output quality but typically act at the prompt level rather than at the topology level. We propose that reliability can be designed as a property of architecture: gating, validation, resource budgets, and information-flow constraints.

3 The Bridge: GRNs and Agents as Open Interfaces in Poly

To relate GRNs and agentic systems, we map both to a common mathematical representation: polynomial functors for interfaces and coalgebras for stateful behavior.

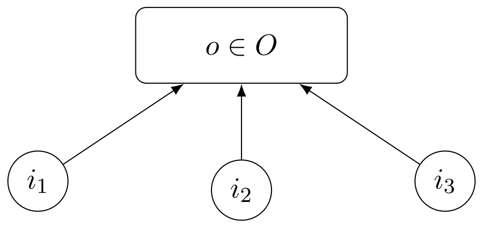
3.1 Preliminaries: Polynomial Functors

Definition 1 (Polynomial Functor)**.** A (finitary) polynomial functor P is an object of **Poly** of the form

$$P(y) \;=\; \sum_{o\in O} y^{I(o)}. \tag{1}$$

Here O is a set of *positions* (outputs). For each $o \in O$, $I(o)$ is the set of *directions* (inputs) enabled/required after emitting o.

Intuitively, a system exposes an output o and then awaits an input $i \in I(o)$ before proceeding.



A “corolla” interface: output then enabled inputs.

Figure 1: A visual mnemonic for a polynomial interface.

3.2 Coalgebras: Stateful Open Systems

Definition 2 (Coalgebra for P)**.** A (deterministic) open dynamical system with interface P is a coalgebra (S, φ) where

$$\varphi : S \to P(S). \tag{2}$$

Expanding $P(S) = \sum_{o\in O} S^{I(o)}$, the structure map yields: (1) a *readout* selecting an output position o, and (2) an *update* selecting next state given an input direction.

This is the level at which both GRNs and agentic systems admit a shared description: state S, interface P, and compositional wiring.

3.3 Two Domain Categories and Two Encodings

We make the scope of the correspondence explicit.

Definition 3 (Interface Categories (Sketch))**.** Let $\mathbf{GRN}_{\text{int}}$ be a category whose objects are *open regulatory modules* with typed signal ports, and whose morphisms are admissible couplings (regulatory connections) that respect those ports. Let $\mathbf{AG}_{\text{int}}$ be a category whose objects are *open agent modules* with typed I/O ports, and whose morphisms are admissible dataflow connections.

Proposition 1 (Encodings into **Poly** (Informal))**.** *There exist interface encodings (functors)*

$$F : \mathbf{GRN}_{\text{int}} \to \mathbf{Poly} \qquad \textit{and} \qquad G : \mathbf{AG}_{\text{int}} \to \mathbf{Poly}$$

that map each module to a polynomial interface and each admissible coupling to a morphism in **Poly**.

Remark 1. We do not claim $\mathbf{GRN}_{\text{int}} \cong \mathbf{AG}_{\text{int}}$ as categories. Instead, we claim both admit a faithful *interface-level* representation in **Poly** under the chosen abstraction (typed ports, open composition, and stateful coalgebra semantics).

3.4 Genes and Agents as Polynomial Interfaces

The original presentation used uniform direction sets, which is too coarse for both domains. We refine the definitions by allowing *position-dependent directions*.

Definition 4 (Gene Interface Object)**.** Fix a gene/module G. Let O_G be the set of possible expression outcomes (e.g., protein isoforms, expression regimes, or "off"), and for each $o \in O_G$ let $I_G(o)$ be the set of regulatory contexts (binding configurations, signal tuples) that are meaningful after o is selected. Define:

$$P_G(y) \;=\; \sum_{o \in O_G} y^{I_G(o)}. \tag{3}$$

Definition 5 (Agent Interface Object)**.** Fix an agent module A. Let O_A be the set of action-positions (e.g., respond to user, call tool κ, request clarification, emit approval token, terminate), and for each $o \in O_A$ let $I_A(o)$ be the expected observation type(s) after taking action o. Define:

$$P_A(y) \;=\; \sum_{o \in O_A} y^{I_A(o)}. \tag{4}$$

This refinement matches reality: different actions induce different expected next observations; and different gene expression regimes correspond to different relevant regulatory inputs.

3.5 Promoters and Schemas as *Validated* Optics (Not "Undefined")

In biology, a promoter gates what signals are relevant to a gene. In software, schemas and context windows gate what data is visible and how it is interpreted. The crucial correction is that mismatches should not be treated as "mathematically undefined"; they should be modeled as *partial* or *effectful* maps with explicit error values.

Definition 6 (Validated Lens (Sketch))**.** A validated lens from global state S to local view V is a pair

$$\text{get} : S \to \mathsf{Either}(\mathsf{Err}, V), \qquad \text{put} : S \times V' \to S,$$

where get may fail with an error value when the promoter/schema does not match.

Remark 2. This aligns with real agent systems: a schema mismatch typically triggers error handling, retries, or safe termination, rather than an "undefined" computation.

3.6 Epigenetics and Memory as State

Both domains are stateful: epigenetic markers bias gene responsiveness without changing DNA, while RAG/history bias agent behavior without changing model weights. In our coalgebra semantics, these correspond to the internal state S.

3.7 Dictionary

Category Concept	Biological Realization (GRN)	Software Realization (Agentic)
Polynomial functor P	Module interface	Agent/module interface
Position set O	Expression outcomes	Action kinds (tool call, reply, stop, ...)
Directions $I(o)$	Regulatory contexts	Expected next observation types
Validated lens	Promoter gating	Schema/context gating with error handling
State S	Epigenetic landscape	Memory/RAG/history
Morphisms	Signal transduction wiring	Dataflow wiring

Table 1: A compositional dictionary at the interface level.

4 A Formal Syntax for Agent Composition: The Operad WAgent

We now define a wiring discipline for agentic systems that supports: (1) static rejection of ill-typed wiring, (2) explicit validation boundaries, and (3) information-flow and capability reasoning.

4.1 Data Types, Integrity Labels, and Capabilities

Definition 7 (Base Data Types)**.** Let $\mathcal{T}$ be a set of base data types, e.g.

$$\mathcal{T} = \{\mathsf{Text},\ \mathsf{JSON}(\Sigma),\ \mathsf{Image},\ \mathsf{ToolCall}(\kappa),\ \mathsf{Error},\ \mathsf{Stop},\ \mathsf{Approval}\}.$$

Definition 8 (Integrity Labels)**.** Let $\mathcal{L} = \{\mathsf{U}, \mathsf{V}, \mathsf{T}\}$ denote integrity levels:

$$\mathsf{U} = \text{untrusted}, \quad \mathsf{V} = \text{validated}, \quad \mathsf{T} = \text{trusted}.$$

We order them $\mathsf{U} \le \mathsf{V} \le \mathsf{T}$ and enforce *integrity-preserving flow*: a wire may connect output label ℓ_{out} to input label ℓ_{in} only if

$$\ell_{\text{out}} \ge \ell_{\text{in}}. \tag{5}$$

Thus untrusted data cannot directly feed a trusted-required port.

Definition 9 (Capabilities / Effects)**.** Let $\mathcal{C}$ be a set of capabilities (effects), e.g.

$$\mathcal{C} = \{\mathsf{ReadFS}, \mathsf{WriteFS}, \mathsf{Net}, \mathsf{ExecCode}, \mathsf{Money}, \mathsf{EmailSend}\}.$$

Each box (agent/module) carries an effect set $\epsilon \subseteq \mathcal{C}$, and composition aggregates effects by union unless restricted by a policy.

4.2 Ports and Well-Formed Wiring

Definition 10 (Port Type)**.** A port carries a *decorated type* (τ, ℓ) where $\tau \in \mathcal{T}$ and $\ell \in \mathcal{L}$. (Effects are box-level annotations, but may be required by specific ports such as execution.)

Definition 11 (Well-Typed Wire)**.** A wire from port (τ_1, ℓ_1) to port (τ_2, ℓ_2) is well-typed when:

$$\tau_1 = \tau_2 \quad \text{and} \quad \ell_1 \geq \ell_2.$$

Remark 3. This is the core distinction between *structural correctness* and semantic correctness. WAgent can prevent structural integration errors (wrong types, wrong trust boundaries) but cannot, by itself, guarantee the semantic truth of a text produced by an LLM.

4.3 Primitive Composition Operations

The operad provides three key wiring primitives.

4.3.1 Parallel Composition ($\otimes$)

Two agents A and B execute in parallel. Parallel composition is safe when they do not race on a shared mutable resource, or when shared resources are mediated by explicit resource boxes (locks, transactions, CRDTs, etc.).

4.3.2 Serial Composition ($\circ$)

The output of A feeds the input of B. In WAgent, ill-typed connections are rejected at design time. This moves *structural mismatch failures* (e.g., feeding text into a JSON port) from runtime to architecture time. It does *not* eliminate semantic hallucinations.

4.3.3 Feedback / Trace Tr

Feedback loops represent agency and self-reference, but require termination governance. Rather than appealing to an undefined "contractive map" property, we introduce an explicit progress measure and budget.

Definition 12 (Guarded Trace (Fuel/Budget))**.** A guarded trace $\mathrm{Tr}_B(A)$ wires feedback through a budget box B that enforces: (1) a maximum number of iterations and/or tokens, and (2) a decreasing ranking/progress measure unless external evidence increases it.

5 Motif 1: Coherent Feed-Forward Loop and Correct Safety Claims

5.1 CFFL as Two-Key Execution

A coherent feed-forward loop has a direct path and an indirect approval path. In agentic systems, we treat the indirect path as a verifier producing an explicit approval token.

5.2 The Correct Failure Probability

We now correct a common but subtle mistake: *an AND gate does not enforce independence.* It enforces conjunction (both signals required). Independence is an assumption about how errors arise.

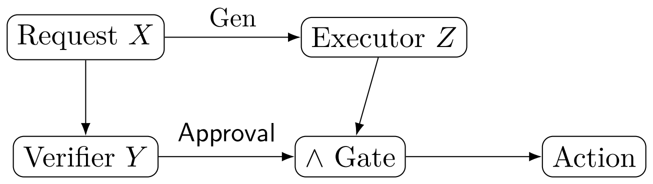


Figure 2: CFFL as two-key execution: executor output is gated by an approval token.

Definition 13 (Error Events)**.** Let E_{gen} be the event that the generator/executor path proposes an unsafe or incorrect action. Let E_{ver} be the event that the verifier *approves* that unsafe/incorrect action (i.e., fails to block it).

Theorem 1 (CFFL Risk Expression)**.** *In a direct serial execution (no gating), the system fails whenever E_{gen} occurs:*

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

In a CFFL with an approval gate, the system fails only when both occur:

$$\mathbb{P}(\text{Fail}_{\text{CFFL}}) = \mathbb{P}(E_{\text{gen}} \cap E_{\text{ver}}) = \mathbb{P}(E_{\text{gen}})\,\mathbb{P}(E_{\text{ver}} \mid E_{\text{gen}}). \tag{6}$$

If the verifier catches generator errors with probability q, i.e. $\mathbb{P}(E_{\text{ver}} \mid E_{\text{gen}}) = 1 - q$, then CFFL reduces failure probability by a factor $(1 - q)$ relative to the direct link.

Proof. Direct execution has no second check, so failure is exactly E_{gen}. In the CFFL, the executor can act only if an approval token is present, so failure requires (1) an unsafe proposal and (2) verifier approval. The probability identity $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B \mid A)$ yields (6). □

Remark 4 (What Topology Guarantees vs What It Does Not)**.** Topology guarantees an *interlock*: unilateral generator output cannot execute without verifier approval. Topology does *not* guarantee $\mathbb{P}(E_{\text{ver}} \mid E_{\text{gen}})$ is small. Reducing that conditional probability requires design choices: model diversity, independent evidence, tool-based verification, and adversarial testing.

5.3 A Topological Guarantee: Two-Key Execution in WAgent

Lemma 1 (Two-Key Execution)**.** *Assume the action sink requires an input port of type* $(\mathsf{Approval}, \mathsf{T})$*. Assume the generator path cannot output* $(\mathsf{Approval}, \mathsf{T})$ *(it outputs at most* $(\mathsf{Text}, \mathsf{U})$ *or* $(\mathsf{ToolCall}(\kappa), \mathsf{V})$*). Then any well-typed wiring diagram that reaches the action sink must include a verifier box that produces* $(\mathsf{Approval}, \mathsf{T})$*.*

Proof. By WAgent typing, the action sink must be fed by a wire with exactly the required type and label. Since the generator cannot produce that port, the only way to satisfy the typing constraints is to include a box whose output includes $(\mathsf{Approval}, \mathsf{T})$, i.e. the verifier. □

6 Motif 2: Quorum Sensing (Consensus & Voting)

Quorum sensing aggregates many weak signals into a robust threshold decision. In agents, this maps to ensembles and voting.

Remark 5 (Correlation Warning)**.** Voting improves reliability when individual errors are not perfectly correlated and when at least a plurality is better than chance. Shared prompts, shared context poisoning, or a shared misconception can make errors highly correlated, producing "confidently wrong" consensus.

Design Implication. To make quorum useful, inject diversity (different models/temperatures/prompts), incorporate tool-grounded checks, and consider robust aggregators (weighted voting, abstention, proof-carrying verifiers).

7 Motif 3: Chaperones as Partial Validators (Not Retractions)

Biological chaperones do not implement a perfect inverse; they attempt folding, detect failures, and route for repair or degradation. The correct software analogue is *partial validation* with explicit error values.

Definition 14 (Chaperone Validator (Kleisli Form))**.** A chaperone is a Kleisli morphism

$$\mathsf{Validate} : (\mathsf{Text}, \mathsf{U}) \to \mathsf{Either}(\mathsf{Err}, (\mathsf{JSON}(\Sigma), \mathsf{V})).$$

It either yields a validated structured value or an error trace used for repair.

Retry & Repair Loop. The generator proposes a structure; the chaperone validates; if invalid, the error is fed back with bounded retries and a budget box.

8 Failure Modes as Pathology

Catastrophic failures often arise not from a single broken component but from dysregulated dynamics.

8.1 Oncology: Infinite Loops as Unchecked Growth

Agentic Pathology. Agents can enter recursive politeness loops or self-debug loops.

Correct Categorical Diagnosis. The problem is not "distance goes to zero" in an unspecified metric space. The problem is absence of a well-founded progress measure and enforced budgets.

Treatment. Introduce a guarded trace with: (1) a step/token budget, and (2) a ranking function $m(S) \in \mathbb{N}$ that must decrease unless new external evidence arrives. If m fails to decrease for k steps, trigger apoptosis (termination) or escalation.

8.2 Autoimmunity: Hallucination Cascades as Integrity Failure

Agentic Pathology. One agent hallucinates; another consumes it from shared memory as if trusted.

Diagnosis. This is an integrity-labeling failure: the system does not distinguish untrusted generated text from validated tool outputs.

Treatment. Use integrity labels in WAgent: tool outputs may be labeled T (trusted), while generated hypotheses remain U unless validated. Require critical decisions to consume V or T inputs only.

8.3 Prion Disease: Prompt Injection as Instruction Contamination

Agentic Pathology. A malicious instruction enters context and influences downstream behavior.

Correct Mechanism Statement. Propagation is primarily due to *context mixing and instruction-following* across trust boundaries, not "pure geometry" alone.

Diagnosis. An information-flow policy is violated: untrusted text affects privileged control decisions.

Treatment.

- **Noninterference by construction:** require privileged actions to be gated by $(\mathsf{Approval}, \mathsf{T})$ tokens and/or tool-grounded evidence.
- **Sanitization layers:** $\mathsf{Sanitize} : (\mathsf{Text}, \mathsf{U}) \to (\mathsf{Text}, \mathsf{U})$ that removes tool-like syntax, policy-violating directives, or embedded prompts (useful but not sufficient alone).
- **Capability least-privilege:** restrict which boxes possess capabilities like Net, $\mathsf{WriteFS}$, or Money; require explicit escalation paths.

8.4 Ischemia: Resource Exhaustion

Agentic Pathology. Token starvation, rate limits, latency spikes.

Diagnosis. Every wiring-diagram operation consumes resources. Static reasoning can bound *structural* fan-out and maximum loop iterations, but dynamic effects (model variability, tool delays) require runtime enforcement.

Treatment. A resource-aware runtime enforces token budgets and timeouts. The compiler can reject architectures that contain *unguarded* trace, but cannot in general compute exact bounds for unconstrained LLM-driven stopping conditions.

9 Discussion: Towards "Epigenetic" Software

9.1 RAG as Digital Methylation

Treat model weights as static "DNA" and RAG/context as epigenetic markers that expose or silence capabilities.

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

where E is an epigenetic state (retrieved documents, memory, policies) shaping behavior without changing W.

9.2 Horizontal Gene Transfer: Dynamic Tool Loading

Tool schemas can be injected at runtime like plasmids:

$$A_{\text{new}} = A_{\text{old}} \otimes \text{ToolSchema}.$$

In WAgent, the new schema adds new typed ports and capabilities, and must be mediated by integrity and effect policies.

9.3 Metabolic Bounds: What Can Be Guaranteed

- **Compiler guarantees:** reject unguarded feedback, enforce type/label constraints, enforce explicit budget boxes on loops, bound fan-out structurally.
- **Runtime guarantees:** enforce token/time/tool-call budgets, enforce capability limits, and trigger safe termination/escalation.

9.4 Endosymbiosis: Neuro-Symbolic Integration

LLMs excel at planning/semantics but are inefficient at exact arithmetic and formal reasoning. Integrating deterministic runtimes (REPLs, solvers) yields a symbiotic system:

$$A_{\text{euk}} = A_{\text{LLM}} \oplus A_{\text{runtime}}.$$

In WAgent, the runtime box has sharply typed ports and produces high-integrity outputs (e.g. $(\mathsf{JSON}(\Sigma), \mathsf{T})$), which can then drive high-stakes decisions.

10 Conclusion

Agentic engineering benefits from biomimetic topology, but safety claims must be precise. Category theory provides an interface-level semantics in which both GRNs and agentic systems are coalgebras for polynomial interfaces composed by wiring diagrams. Biological motifs become reusable patterns: persistence gates (CFFL), ensemble consensus (quorum), structural validators (chaperones), and immune-like trust boundaries.

We introduced WAgent as a typed operadic syntax that supports: (i) structural correctness by type checking, (ii) explicit validation via partial/effectful optics, (iii) information-flow constraints via integrity labels, and (iv) security/resource governance via capability and budget discipline.

A central correction is that topology provides interlocks, not independence. To reduce correlated failure, we must combine topology with diversity and tool-grounded verification.

References

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

[2] Brendan Fong and David I Spivak. *Seven sketches in compositionality: An invitation to applied category theory*. Cambridge University Press, 2019.

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

[4] 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.

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

[6] 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.

[7] 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`.