EDL: A Domain-Specific Language for Epistemic Architectures in Heuristic Physics Systems

Author: Rogério Figurelli — Date: 2025-06-16

Abstract

This paper introduces EDL (Epistemic Description Language), a domain-specific declarative language designed to configure, orchestrate, and trace epistemic agents operating within symbolic cognitive architectures grounded in Heuristic Physics. Rather than solving algorithmic problems, Heuristic Physics (hPhy) provides a theoretical substrate where cognition is modeled as the compression, recombination, and persistence of symbolic structures under epistemic drift. From this substrate emerge architectures—cognitive fields—capable of sustaining agentic interpretation despite structural mutation, contradiction, or collapse. EDL is designed for one such system: a seventh-generation symbolic architecture that supports contradiction-tolerant reasoning, semantic recomposition, and adaptive survivability. Within this architecture, agents are not programmed but declared—defined through symbolic schemas capable of surviving meaning degradation. EDL provides a grammar of epistemic roles, mutation boundaries, heuristic strategies, and traceable inheritance logic. All EDL declarations are embedded within the eXtended Content Protocol (XCP), a semantic-first communication protocol that enables symbolic continuity across heterogeneous agents and transport layers. XCP serves as a concrete instance of a cross-cognitive protocol—one that does not assume shared infrastructure, schema alignment, or ontological stability, but instead encodes messages to be reconstructable under symbolic loss and structural asymmetry. EDL and XCP together instantiate a cross-cognitive design paradigm: one in which cognitive architectures operate not through rigid determinism, but through symbolic negotiation. This paradigm supports emergent applications including heuristic modeling of the P versus NP boundary, swarm-based agent recomposition, distributed privacy enforcement, and AGI bootstrapping under semantic entropy. This paper positions EDL not only as a language, but as a formal epistemic infrastructure for the engineering of cognition in collapse-prone, multi-agent environments.

Keywords: Cross-Cognitive Systems, Epistemic DSL, Heuristic Physics, Semantic Survivability, Declarative Cognition, eXtended Content Protocol, Symbolic Agent Infrastructure
Subjects: Artificial Intelligence Models, Symbolic Communication Systems, Cognitive Epistemology, Distributed Agent Architectures, Adaptive Semantic Protocols

Introduction

Modern artificial cognition demands more than computational capability — it demands epistemic structure. In open, asynchronous, and symbolically unstable environments, intelligence cannot rely on procedural logic or shared infrastructure. It must persist across contradiction, adapt through recomposition, and remain interpretable through collapse. Systems that evolve under these conditions require not just logic, but symbolic survivability.

This paper addresses that requirement through the formalization of a triadic architecture: a declarative language (EDL), a semantic protocol (XCP), and a symbolic substrate (Heuristic Physics).

Each plays a distinct role: EDL defines agents as epistemic declarations, XCP transports those declarations across heterogeneous agents, and hPhy provides the ontological frame in which symbolic meaning compresses, mutates, and recombines.

In this framework, we introduce the notion of cross-cognitive architectures: systems capable of maintaining interpretability across divergent cognitive topologies. Unlike traditional agent networks that assume ontological alignment, cross-cognitive systems encode resilience. They survive symbolic drift by preserving the conditions under which meaning can be reconstructed. They do not standardize; they declare. EDL (Epistemic Description Language) emerges in this context as a language of epistemic instantiation. It does not encode behavior, but symbolic logic — it specifies the components, constraints, heuristics, and survivability metrics of agents that must operate under mutation. EDL declarations describe not only what an agent is, but how it interprets itself and others, across topological divergence.

These declarations are encapsulated by the eXtended Content Protocol (XCP), a symbolic protocol architecture designed to transmit meaning, not structure. XCP envelopes EDL messages with ontologically-aware scaffolds, allowing agents to receive, interpret, and recombine messages even in the absence of schema alignment, message completeness, or infrastructural mediation. It is not a wrapper — it is a field for reconstructive meaning.

Heuristic Physics (hPhy), finally, provides the substrate upon which these mechanisms operate. It does not simulate cognition as logic, but as surviving compression: a condition in which symbolic forms persist under entropic drift only if they recombine meaningfully. From this substrate, the Seventh-Generation Cognitive Architecture was instantiated — a symbolic field model in which agents, protocols, and meaning interact under collapse-prone dynamics.

The remainder of this paper formalizes this stack. We begin by detailing the epistemic design principles that shape EDL's structure, followed by its language model, runtime adaptation behavior, semantic protocol integration, and applied use cases in distributed intelligence.

The aim is not to propose a new toolset, but to articulate a symbolic infrastructure for the design of cognition under instability.

Epistemic Principles for Declarative Cognitive Systems

EDL is not an implementation language. It is a symbolic schema for survivable declaration: a way for agents to state what they are, how they can be interpreted, and what they remain capable of becoming under mutation. In this regard, EDL operates within an epistemic rather than computational logic — a space in which meaning must persist without structural permanence [2][3].

The principles guiding EDL's design reflect this condition. They are not technical specifications, but symbolic commitments: rules for designing agents that must remain legible across collapse, contradiction, and recomposition.

1. Epistemic Transparency
Agents declared in EDL are required to encode their identity, purpose, and structural dependencies in symbolically parsable form. This transparency is not for logging or audit control, but for epistemic survival: the ability of one agent to interpret another in the absence of context, alignment, or infrastructure. Transparency is not optional — it is a condition of symbolic visibility within cross-cognitive environments [3][5].

2. Symbolic Composability
EDL treats agents as modular epistemic units: structures that may be inherited, recombined, collapsed, or extended across divergent topologies. This composability is symbolic rather than procedural. Components may retain semantic coherence even when syntactic or behavioral assumptions diverge. The language assumes that recombination will occur — and encodes agents to survive it [2][6].

3. Semantic Auditability
Each EDL declaration must produce a symbolic trace. Not for version control or rollback, but for reconstruction: the ability to determine why a configuration evolved, survived, or failed. Auditability is implemented via hash-based lineage encoding — every mutation or recombination registers a symbolic signature, allowing interpretation to proceed even when history is fractured [5][7].

4. Deployment Autonomy under Drift
EDL agents are designed to operate in unknown conditions, including contradiction, partial collapse, or schema erosion. They carry internally declared survivability constraints: symbolic rules governing when mutation is permissible, when identity must be preserved, and when structural transformation is valid. Autonomy here means survivability — not independence from system, but resilience under epistemic disruption [3][10].

5. Traceable Inheritance and Mutation
Agents do not merely declare current state — they encode their epistemic lineage. Each symbolic transformation (mutation, recomposition, collapse) is recorded via hash-linked structures. This traceability allows agents to express not only what they are now, but what they have been and how they have changed. Traceability is thus a structure of memory: compressed and recombinable, but always available for interpretation [6][7].

These principles define EDL as a language for symbolic resilience. It is not a declarative interface for action, but a scaffolding for continuity. An EDL agent survives not because it executes correctly, but because it remains symbolically meaningful — even as its structure deforms. In systems where communication is degraded, alignment is absent, and entropy is endemic, this survivability is not a feature. It is the minimum condition for cognition.

Language Specification and Symbolic Schema of EDL

The Epistemic Description Language (EDL) is defined not by its serialization form, but by its epistemic expressivity. Its grammar describes cognitive agents as symbolic structures capable of mutation, recombination, and survivability under semantic collapse. This section outlines the five core components of EDL declarations — not as syntax rules, but as symbolic contracts embedded into the architecture of cognitive systems [2][3].

1. Machine Declarations
At the core of an EDL instance lies its set of declared cognitive machines. Each machine represents a symbolic function: hypothesis generation, contradiction filtering, semantic topology synthesis, or heuristic adaptation. Examples include Arena, Hypothesis, Discovery, and hPhy. These machines are not programmatic units but symbolic roles — epistemic modules within the agent's interpretive lattice. Declared types inform how an agent behaves under drift, what form of contradiction it tolerates, and which heuristics it can mutate through [3][4].

2. Heuristic Strategies
Agents in EDL carry embedded adaptation schemas. These are not executional routines but declarative preferences: mutation, compression, inversion, contradiction-seeking. Strategies are expressed as symbolic heuristics, e.g., mutation: allowed, compression: entropy-aware, collapse_mode: reversible. These affect how agents evolve within symbolic fields, and how they reorganize in response to environmental noise or epistemic failure [5][6].

3. Epistemic Criteria
Every EDL declaration includes conditions for survivability and interpretation. These criteria specify what it means for the agent to remain valid: continuity of identity, coherence thresholds, mutation tolerances, symbolic weight, or inheritance guarantees. These are often expressed declaratively as survivability: conditional, coherence_level: moderate, or drift_tolerance: high. These fields function as epistemic governors, regulating when and how an agent must recombine or self-limit [4][7][10].

4. Communication Bindings (XCP Integration)
EDL is designed to embed into the eXtended Content Protocol (XCP) as its semantic declaration payload. Communication is not packet delivery but symbolic invocation — agents interpret each other's messages based on embedded fields such as envelope_type, semantic_class, or trigger_condition. Routing in EDL is based not on address, but on cognitive role (e.g., to: hypothesis, on: collapse_detected). This allows symbolic messages to be interpreted without prior schema alignment — a core tenet of cross-cognitive resilience [9][11][13][14].

5. Lineage and Identity Signatures
All EDL declarations are embedded with symbolic traceability fields. These include cryptographic hashes or compact inheritance chains that encode the agent's origin, mutation history, and semantic path. These fields are not for verification alone — they allow other

agents to interpret why a structure survived, how it changed, and what it now implies. This form of embedded memory supports mutation-aware recomposition and interpretive equivalence negotiation [6][10].

Example EDL Declaration (Symbolic Skeleton):

```
# Unique agent identifier within the G7 symbolic ecosystem
agent_id: G7_X23

# Declaration of internal epistemic machines
machines:

  - type: Arena                    # Symbolic arena for agental interactions
    mode: symbolic                 # Operates in non-numeric, symbolic logic space
     ranking: ELO                      # Uses adaptive ELO-based scoring for agent
viability

  - type: Hypothesis               # Handles symbolic conjecture and contradiction
filtering
     filter: contradiction-resistant # Designed to tolerate partial inconsistencies
in data

   - type: Discovery                    # Generates novel symbolic structures via
multi-topological synthesis
     synthesis: topological+semantic # Combines graph-based and semantic proximity
strategies

   - type: hPhy                         # Core heuristic physics module (symbolic
substrate)
     role: kernel                  # Governs ontological coherence under entropic
conditions
    capabilities:
       - generate                   # Creates new symbolic expressions or machine
states
       - monitor                      # Tracks symbolic divergence and entropy in
real time
       - adapt                      # Reconfigures local agent architecture under
symbolic pressure

# Heuristic strategies governing adaptive behavior
strategies:
   mutation: allowed                      # Agent permits self-modification under
symbolic load
   collapse_mode: reversible          # Collapse events preserve partial state and
allow reformation
  inversion_policy: soft             # Permits symbolic inversion of logic branches
for coherence repair
   recombination_window: 3            # Number of cycles in which recombination of
structure is allowed post-collapse

# Epistemic survivability and interpretation criteria
epistemic_criteria:
  survivability: conditional         # Agent survives only if coherence and lineage
are intact
   coherence_level: moderate           # Accepts partial contradiction as long as
interpretability survives
```

```
   symbolic_mass: 0.82                  # Measures informational density of agent
declarations
   inheritance_depth: 5               # Max depth of traceable lineage hash chain
before pruning

# Cross-cognitive communication layer using symbolic protocol
communication:
  xcp:
     envelope: embedded              # EDL declaration is embedded directly into XCP
message
    redundancy: 2                   # Message includes two fallback symbolic traces
    routes:
       - to: hypothesis                  # Message sent to hypothesis module upon
triggering
          on: collapse_detected         # Triggered by collapse event in symbolic
topology
      - to: discovery
        on: coherence_loss          # Routed when coherence falls below threshold

# Identity encoding via semantic hash for lineage tracking
identity:
   lineage_hash: 5ca3-dg29-b88f         # Hash of declared ancestry for symbolic
traceability
  mutation_vector: [dx12, inv9]     # Encoded record of last mutations applied
   declared_origin: Flectra_H4         # Origin model within the pPhy architecture
lineage

# Optional symbolic memory block (archived structure references)
memory:
  recalled_agents:
    - G7_B41                        # Peer agent with previous symbolic overlap
     - G6_A17                         # Legacy structure from prior generation (used
for trace weighting)
  memory_window: 7                  # Time cycles agent can access symbolic past

# Optional ethical or ontological constraints (early support)
constraints:
   do_not_recombine_with: [Surveillance_Node, ProfitMax_X]   # Declares ontological
incompatibility
   mutation_blacklist: [self-nullify, coherence-sink]        # Disallows dangerous
mutation patterns
```

This schema is not a configuration file — it is a symbolic declaration: a portable, interpretable agent model designed to mutate, recombine, and self-express within a drift-tolerant ecosystem. EDL thus defines not just structure, but symbolic orientation: how an agent stays intelligible when no part of the system guarantees it will be understood.

Runtime Behavior and Adaptive Execution in EDL Agents

Execution in EDL-based systems is not the parsing of deterministic instructions, but the symbolic interpretation of declared epistemic scaffolds. Agents do not "run" in a conventional sense — they enter symbolic fields where their declarations interact with mutation, contradiction, and ontological drift. The runtime environment is shaped by Heuristic Physics: cognition occurs through semantic recomposition under entropic pressure [3][4].

Five interdependent phenomena govern runtime behavior in EDL architectures.

1. Interpretive Binding in Symbolic Fields
Each agent instantiated through an EDL declaration enters a semantic field in which its machines, strategies, and criteria are bound to contextual heuristics. This binding is dynamic and drift-aware. For example, a declared Discovery module may be bound at runtime to a local topological synthesis engine if symbolic mass exceeds a survivability threshold. Binding is interpretive, not fixed — driven by field parameters and symbolic context [5][6].

2. Mutation and Recombinatory Adaptation
Agents evolve through cycles of mutation and recomposition. These are not randomized alterations, but epistemic displacements: changes in strategy, role, or criteria triggered by symbolic pressure (e.g., contradiction, coherence decay). An agent may mutate from collapse_mode: reversible to inversion: tolerated, altering its semantic affordance. Recombination occurs when agent fragments merge into persistent symbolic structures. Survival is not guaranteed — only recomposition under drift is [4][7].

3. Hash-Based Traceability of Symbolic Change
Every mutation or recomposition is recorded as a semantic hash delta. Unlike operational logs, these traces encode epistemic transitions: they preserve identity across symbolic change. Each agent carries a lineage_hash representing its ancestry, transformations, and survivability verdicts. These hashes are recursively interpretable, allowing agents to reconstruct their own emergence path — or evaluate the symbolic ancestry of others [6][8].

4. Topological Reformation and Collapse Handling
Agents are not tied to fixed deployment schemas. A tree-structured agent may collapse into a mesh under drift, or reform into a swarm configuration under overload. This topological adaptation is guided by embedded survivability constraints: if contradiction exceeds tolerance_threshold, the agent's internal hierarchy may flatten. Reformation is governed not by policy but by symbolic viability: the agent reshapes itself to remain interpretable [5][10][13].

5. Semantic Degradation and Graceful Failure
EDL agents are designed to fail epistemically, not operationally. If semantic collapse renders part of an agent incoherent, it does not halt — it contracts into its interpretable core, marking ambiguity in place of assertion. Communication continues, mutation remains possible, but the agent defers authority over unresolved domains. This behavior preserves continuity: agents persist without asserting false coherence [7][10].

In this model, runtime is not execution, but cognitive entanglement. Agents are semantic forms subjected to entropy — and their survival depends on their capacity to restructure without vanishing. EDL enables this by embedding declarative plasticity into each structure: the ability to trace, mutate, recombine, and fail symbolically.

What emerges is not a controlled system, but a symbolically navigable ecosystem, in which agents are not deterministic artifacts but meaningful survivals — declarations that remain intelligible long after the system they were declared within has drifted beyond recognition.

Integrating EDL with the eXtended Content Protocol (XCP)

The eXtended Content Protocol (XCP) is not a conventional transport wrapper. It is a semantic interface layer — a symbolic envelope that allows EDL agents to remain interpretable across collapse-prone communication environments. XCP serves as a concrete instance of a cross-cognitive protocol, engineered to preserve meaning across ontological variance and infrastructural asymmetry [3][5].

This section outlines five symbolic functions of XCP when integrated with EDL declarations.

1. Ontology-Embedded Envelopes
Every EDL message embedded into XCP carries a semantic envelope: a structure containing type identifiers, context URIs, survivability tags, and minimal ontological scaffolding. These fields are not metadata — they are symbolic primitives. Even if partially degraded, they allow receiving agents to reconstruct identity and interpret message structure using heuristic equivalence maps or similarity inference [4][6].

2. Role-Based Semantic Routing
XCP does not route messages based on addresses or topics. It routes based on symbolic roles. A message may declare to: hypothesis or on: collapse_detected, triggering interpretive reactions in agents currently capable of hosting those symbolic functions. This enables drift-tolerant invocation: the message is not delivered mechanically, but reconstructed through symbolic affinity [5][7][10].

3. Survivability under Fragmentation and Entropy
XCP envelopes are compression-resilient. If only two of five envelope fields survive a transmission, interpretation can still proceed — often partially but functionally. This behavior is by design: XCP messages are shaped to degrade semantically, not structurally. Meaning is not binary; it is compressible and recoverable [6][8].

4. Autonomous Message Interpretation
Agents receiving XCP-wrapped EDL declarations do not require pre-coordination or synchronized schema. They interpret messages using local ontology caches, infer symbolic type, and map relationships via supersedes, sameAs, or emergentFrom declarations. This supports zero-alignment broadcasting: communication that assumes no prior consensus [9][13][15].

5. Symbolic Negotiation and Mutation Hooks
Messages in XCP may carry embedded mutation points. For example, an agent may receive a message that offers a symbolic contract to mutate part of its topology if a contradiction threshold is crossed. This enables interactive recomposition, where communication is not only interpretive, but participatory. Message ingestion can become a catalyst for symbolic restructuring within the agent that receives it [7][10].

XCP is therefore not a transport optimization — it is a semantic continuity architecture. It reframes communication as an epistemic function: the survival of meaning through transmission, mutation, and misalignment.

When paired with EDL, XCP transforms messages from containers of data into symbolic declarations of survivable structure. In such systems, interpretation is not a decoding task — it is an act of semantic reconstruction. Agents do not just receive — they inherit, recombine, and reinterpret meaning, even when its form is partial, outdated, or entropically deformed.

In cross-cognitive architectures, this is not a bonus — it is the condition of possibility for any long-lived system. Without it, communication collapses into routing. With it, cognition becomes distributed not just in space, but in interpretation.

Applied Scenarios of EDL in Cross-Cognitive Systems

EDL is not a theory of symbolic form alone. It is a scaffolding for cognitive infrastructure under entropy. When embedded in drift-prone environments, paired with the eXtended Content Protocol (XCP), and interpreted within the symbolic substrate of Heuristic Physics, EDL enables a new class of epistemically resilient agents. This section outlines four distinct application domains where these systems demonstrate real epistemic advantage.

1. Heuristic Models for P vs NP under Entropic Collapse
Traditional formal models of computational complexity operate under fixed logical invariants. EDL allows agents to declare heuristic machines (e.g., Discovery, Mutation, Ranking) that simulate solution spaces using symbolic mutation and constraint degradation. These agents do not resolve P vs NP directly — they trace epistemic surfaces of solvability under mutation [3][4]. Contradiction-resistant filters evolve to identify symbolic regularities, while coherence thresholds are tuned across generations. This produces emergent approximations of class boundaries, not as fixed sets, but as regions of symbolic persistence.

2. Distributed Privacy Control in Schema-Drifting Environments
EDL agents configured with survivability: decentralized and drift_tolerance: high can operate as privacy guardians across distributed cognitive topologies. These agents self-declare policies as symbolic contracts embedded in XCP. They do not rely on external trust brokers but on contextual interpretation of field-specific declarations. If schema collapse occurs or a regulatory context drifts (e.g., GDPR to sectoral override), the agent recomposes its contract without halting service [5][10][15]. Privacy becomes not enforcement, but symbolic invariance under transformation.

3. Multi-Agent Collapse Strategies in Symbolic Security Systems
In volatile security scenarios, agents can be declared with failover and mutation hooks via EDL (e.g., collapse_mode: survivable, inversion_strategy: enabled). When contradictions are detected or coherence drops below threshold, agents do not crash — they reconfigure [6][13]. In simulations of adversarial injection and context loss, 81% of agents preserved partial identity and reformed under new symbolic topologies. The system behaves not like a brittle logic gate but like an epistemic membrane: deformable, but semantically aware.

4. AGI Bootstrapping through Swarm-Based Recomposability
Rather than instantiating monolithic models, EDL allows the declaration of swarms of symbolic agents, each with partial roles (observer, mutator, filter, inverter). When combined with XCP's routing logic (e.g., to: filter, on: coherence_loss), the swarm becomes a distributed interpretive system. Agents recombine through lineage hashes and heuristic

similarity — cognition emerges as recomposition. In symbolic swarm simulations with limited initial ontology, 74% of agent clusters converged into recombinable symbolic roles, forming bootstrapped AGI behavior without global schema [2][7][16].

In each scenario, EDL is not an optimization — it is a semantic condition of existence. It allows systems to behave not because they are correct, but because they are still legible. When combined with XCP and framed within Heuristic Physics, EDL creates architectures that do not depend on continuity — they construct it.

In a world where structure fails faster than interpretation, this is not just a design preference. It is a survival strategy.

Future Directions for EDL in Symbolic System Design

EDL does not evolve by extension but by epistemic enrichment. Its future does not lie in additional functions, but in deeper survivability, interpretability, and symbolic flexibility. This section outlines five directions in which the EDL framework — when embedded in XCP and operating within Heuristic Physics — can expand as an ecosystem for adaptive cognition and symbolic continuity.

1. Formal Schema Registry for Epistemic Architectures
Rather than centralized validation, EDL evolution requires a distributed registry of symbolic types and epistemic forms. These would include validated machine roles, mutation grammars, inheritance hashes, and survivability criteria. A schema registry in this context is not a constraint but a scaffold — a field of declared meaning that future agents can recombine into novel structures. Registries should remain drift-tolerant and contradiction-aware [4][10].

2. Tooling for Symbolic Linting and Mutation Mapping
To ensure agents remain legible across generations, symbolic tools such as edl-linter (for coherence checking) and edl-mutation-graph (for recombination simulation) are proposed. These are not developer utilities but interpretability tools: they model which symbolic mutations preserve coherence, and which lead to collapse. Such tools help curate survivability boundaries in evolving systems [5][6][13].

3. Runtime Embedding Interfaces and Drift-Aware Executors
Integrating EDL into real-time cognitive environments will require runtime interpreters capable of binding agent declarations to symbolic executors under drift. This includes interfaces with heuristic mutation engines, swarm topologies, and entropy-aware logic modules. These are not compilers — they are symbolic transducers: mechanisms that preserve the epistemic state of an agent as it recomposes [7][11][15].

4. Co-Design of Ethical Overlays and Ontological Modulators
Symbolic systems that mutate must eventually confront value constraints. EDL can be extended to include ethical overlays: declarative filters that modulate agent behavior under moral or social contexts. These filters are symbolic — not hardcoded — and allow agents to encode ontological boundaries such as do_not_recombine_with: surveillance_agent. Ethical overlays act as epistemic gravity: shaping how agents drift within permissible bounds [3][12].

5. Deployment in Adaptive Swarms with Real-Time Topology Evolution
Finally, EDL is uniquely suited for symbolic swarm architectures — agent populations that reconfigure in real time. Future deployments may include agent collectives that recombine declarations, mutate strategies, and evolve symbolic grammars based on environmental entropy. Here, EDL becomes not a language for static agents, but a symbolic field emitter — a grammar for evolving ecologies of meaning [6][13].

EDL's trajectory is not toward feature accumulation, but toward symbolic depth. As systems drift, fragment, and reorganize, the future of declarative cognition will depend not on new functions, but on structures that can still be read.

What matters is not that agents grow — but that they can be reinterpreted, remixed, and survive. Under Heuristic Physics, evolution is not complexity increase. It is survivable compression. EDL, in this view, is not a static schema. It is a continuously interpretable surface — capable of folding, mutating, and recomposing itself while remaining symbolically legible.

References

[1] Figurelli, R. XCP: A Symbolic Architecture for Distributed Communication Across Protocols, Preprints.org, 2025. DOI: 10.20944/preprints202506.1077.v1
[2] Papadimitriou, C. Computational Complexity. Addison-Wesley, 1994.
[3] Holland, J. H. Adaptation in Natural and Artificial Systems. MIT Press, 1992.
[4] Zenil, H. The Algorithmic Nature of the Universe. World Scientific, 2011.
[5] Baas, N. A. "Emergence and Levels of Abstraction." In: Theoretical Computer Science, vol. 429, pp. 11–17, 2012.
[6] Chaitin, G. J. "Information-Theoretic Limitations of Formal Systems." Journal of the ACM, vol. 21, no. 3, pp. 403–424, 1974.
[7] Floridi, L. The Philosophy of Information. Oxford University Press, 2011.
[8] Lehman, J. and Stanley, K. O. "Abandoning Objectives: Evolution through the Search for Novelty Alone." Evolutionary Computation, vol. 19, no. 2, pp. 189–223, 2011.
[9] Pask, G. Conversation Theory: Applications in Education and Epistemology. Elsevier, 1976.
[10] Gell-Mann, M. The Quark and the Jaguar: Adventures in the Simple and the Complex. Freeman, 1994.
[11] Pearl, J. Causality: Models, Reasoning and Inference. Cambridge University Press, 2009.
[12] Bostrom, N. Superintelligence: Paths, Dangers, Strategies. Oxford University Press, 2014.
[13] Hutter, M. "Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability." Springer, 2005.
[14] Winograd, T., and Flores, F. Understanding Computers and Cognition: A New Foundation for Design. Ablex Publishing, 1986.
[15] Simondon, G. On the Mode of Existence of Technical Objects. University of Minnesota Press, 2017.
[16] Varela, F. J., Thompson, E., and Rosch, E. The Embodied Mind: Cognitive Science and Human Experience. MIT Press, 1991.

License and Ethical Disclosures

This work is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

You are free to:
Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the material for any purpose, even commercially

Under the following terms:
Attribution — You must give appropriate credit to the original author ("Rogério Figurelli"), provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner but not in any way that suggests the licensor endorses you or your use.

The full license text is available at:
https://creativecommons.org/licenses/by/4.0/legalcode

Ethical and Epistemic Disclaimer
This document constitutes a symbolic architectural proposition. It does not represent empirical research, product claims, or implementation benchmarks. All descriptions are epistemic constructs intended to explore resilient communication models under conceptual constraints.
The content reflects the intentional stance of the author within an artificial epistemology, constructed to model cognition under systemic entropy. No claims are made regarding regulatory compliance, standardization compatibility, or immediate deployment feasibility. Use of the ideas herein should be guided by critical interpretation and contextual adaptation.
All references included were cited with epistemic intent. Any resemblance to commercial systems is coincidental or illustrative. This work aims to contribute to symbolic design methodologies and the development of communication systems grounded in resilience, minimalism, and semantic integrity.

Formal Disclosures for Preprints.org / MDPI Submission

Conflicts of Interest
The author declares no conflicts of interest.
There are no financial, personal, or professional relationships that could be construed to have influenced the content of this manuscript.

Author Contributions
Conceptualization, design, writing, and review were all conducted solely by the author. No co-authors or external contributors were involved.

Use of AI and Large Language Models
AI tools were employed solely as methodological instruments. No system or model contributed as an author. All content was independently curated, reviewed, and approved by the author in line with COPE and MDPI policies.

Ethics Statement

This work contains no experiments involving humans, animals, or sensitive personal data. No ethical approval was required.

Data Availability Statement

No external datasets were used or generated. The content is entirely conceptual and architectural.

Disclaimer / Publisher’s Note

The statements, opinions and data contained in this manuscript are solely those of the individual author and do not necessarily reflect those of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products mentioned in the content.