ISSN 2395-8618

Systemic Knowledge Compilation: Operationalizing n-ary Hypergraphs for Local LLM Reasoning

Christian Efraín Maldonado-Sifuentes*, Samuel Solís-Gamboa, Mariano Vargas-Santiago, Maria del Carmen Heras-Sanchez, Luis Lechuga-Gutierrez

***Abstract*—As Large Language Models (LLMs) continue to scale, the transition from monolithic, opaque information storage to structured, queryable cognitive architectures becomes imperative for the development of Proto-AGI systems. Current Knowledge Graphs, restricted by binary edge constraints, fail to capture the multidimensional semantic interactions inherent in complex domains. This paper introduces a computationally sustainable architecture for the construction of n-ary mathematical hyperontologies, operationalized through a modular, local extraction pipeline. We present `intuyt:macro`, a custom fine-tuned Mixture of Experts (MoE) model with 3 billion active parameters, designed specifically for domain-specialized topological extraction in Mexican Spanish and indigenous language corpora. By implementing a greedy, token-bounded partitioning algorithm and deterministic JSON sanitization, we enable the compilation of hyperontologies exceeding 20,000 nodes on edge hardware, while bypassing cloud-based latency and privacy risks. Finally, we propose a WebGL-based visualization framework employing selective rendering and topological bounding ($N_{limit} = 300$) to enable fluid, real-time exploration of hyper-dimensional memory structures. Our results demonstrate that local, iterative cognitive loops achieve superior structural adherence and cultural fidelity compared to general-purpose cloud-based APIs, establishing a viable pathway toward efficient, persistent, and self-governing Proto-AGI architectures.**

***Index Terms*—Proto-AGI, mathematical hyperontologies, local LLMs, topological reification, LogicPoison, computational sustainability.**

I. Introduction

The evolution of artificial intelligence is currently traversing a critical transitional phase from Artificial Narrow Intelligence (ANI) towards Artificial General Intelligence (AGI). This intermediate paradigm is conceptualized as Proto-AGI: a systemic architecture where specialized, localized models are integrated with persistent memory, tool-use capabilities, and continuous feedback loops to emulate general cognitive behaviors [1]–[3]. Unlike standard monolithic Large Language Models (LLMs) that act as static stateless functions, a Proto-AGI operates as an advanced, continuous conversational agent capable of iterative logical deduction [4], [5].

Manuscript received on 21/01/2025, accepted for publication on 20/03/2025. Corresponding author is Christian Efraín Maldonado-Sifuentes (christianemaldonadomti@gmail.com).

Christian Efraín Maldonado-Sifuentes, Samuel Solís-Gamboa, Mariano Vargas-Santiago, Luis Lechuga-Gutierrez are with Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), Mexico City, Mexico.

Maria del Carmen Heras-Sanchez is with Universidad de Sonora (UNISON), Mexico.

To formally understand and construct a Proto-AGI, we must abandon linear pipeline architectures and adopt the principles of General Systems Theory (GST) [6]. GST postulates that complex entities are not merely aggregates of their parts, but are defined by the intricate, non-linear interactions within a holistic system. In the context of Proto-AGI, the cognitive capacity emerges from the continuous interaction between the generative language model, the semantic extraction engines, and the underlying mathematical representation of its knowledge base.

Currently, Retrieval-Augmented Generation (RAG) and standard Knowledge Graphs (KGs) are widely employed to act as the memory subsystem of these agents [7]. However, standard KGs fail to satisfy the systemic complexity required by GST.We can divide bodies of knowledge into entities and relations, and traditional graphs are highly efficient at modeling this knowledge by representing it through binary relations between two distinct entities. However, there are certain areas of technology, science, and knowledge whose components are best described as relations among more than two entities. This occurs either because they are the result of interactions among heterogeneous elements, or because multiple binary relations can be represented by a single multidiadic relation. For instance, in social networks, there are phenomena described as simultaneous interactions among different elements from distinct groups. Another example is the collaboration among scientists when co-authoring publications. Similar modelable relations are found in transportation networks, which can be analyzed through the interactions among different areas via the transit services that connect them, or in cognitive brain functions, where specific processes require a complex interaction among distinct brain regions [8]. In more fundamental fields, we find this structural representation in certain chemical substances, genomic sequences, as well as in VLSI design [9].

In classical Graph Theory, a knowledge graph is defined as a tuple $G = (V, E)$, where the edges $e \in E$ representing relations are strictly binary subsets of the Cartesian product of vertices, such that $E \subseteq V \times V$. While computationally convenient, this binary constraint forces multidimensional real-world semantics into flat, lossy representations, rendering the system highly vulnerable to topological exploits and logical contradictions (e.g., LogicPoison attacks) [10].

To satisfy the multi-causal interactions demanded by GST, our architecture replaces standard KGs with Mathematical

ISSN 2395-8618

Hyperontologies based on Set Theory and Hypergraph Theory. A hyperontology is modeled as a hypergraph $H = (X, E)$, where the relation $e \in E$ is not restricted to two nodes, but is an element of the power set of X ($E \subseteq \mathcal{P}(X) \setminus \{\emptyset\}$) [11]. This n-ary structural capacity allows a single logical relation to bind an arbitrary set of ontological concepts simultaneously, preserving the true systemic nature of the knowledge.

Building upon our foundational proofs of concept for Proto-AGI architectures [1], [12], this paper introduces a computationally optimized LLM-Embedding Collaborative Framework. By applying previous findings in context compression [13], our system integrates local LLMs with a dynamic topological reification engine. Utilizing hyper-optimized data serialization formats, the framework extracts, translates, and renders localized n-ary semantic subgraphs in real-time, effectively deploying a scalable Proto-AGI loop in memory-constrained environments.

The main contributions of this work are three-fold:

- We formalize a Proto-AGI architecture through the lens of GST, utilizing local, resource-efficient LLMs for persistent topological reasoning.
- We introduce a dynamic hyperontology engine grounded in Set Theory and n-ary hypergraphs to mitigate logical poisoning and prevent semantic degradation.
- We demonstrate a scalable visual and operational framework that handles large-scale hyperontologies ($> 20,000$ nodes) in client environments through selective subgraph extraction and topological reification.

II. Related Work

A. Proto-AGI and General Systems Theory

The theoretical pursuit of AGI has recently shifted towards modular, systemic approaches. Recent discourse highlights Proto-AGI as an "Expanded Narrow Intelligence," where autonomous loops, memory persistence, and dynamic tool orchestration simulate a cohesive cognitive entity [2], [5]. GST provides the mathematical and philosophical vocabulary for this transition, establishing that the cognitive robustness of a Proto-AGI depends heavily on the systemic equilibrium between its internal knowledge representation and its generative output [3], [6].

B. Logical Pluralism and n-ary Hyperontologies

The conceptual foundation of our knowledge architecture is deeply rooted in logical pluralism, a paradigm championed by Carnap and formally expanded by Goguen [11]. Pluralism posits that complex systemic domains require heterogeneous logical systems to represent knowledge accurately.

Hypergraphs are not rigidly defined; on the contrary, they can be mathematically adapted to become compatible with the representation of diverse types of relations. For instance, depending on the systemic requirements, we might need some hyperedges to be empty, the vertex set to be empty, certain vertices to remain isolated (not belonging to any hyperedge), or different hyperedges to incident on the exact same number of hypervertices [9]. Regardless of the strict definition chosen or the specific multidimensional relations we wish to represent, the extraction and assembly architecture we present remains valid and functionally robust.

This is operationalized by the Distributed Ontology Language (DOL), which facilitates interoperability across diverse logical frameworks [14]. Within DOL, a hyperontology allows multiple domain-specific ontologies to coexist. By moving from classical graph theory to n-ary hypergraphs, our architecture utilizes Set Theory to model complex dependencies where a single event or concept C_k can simultaneously modify a set of targets $\{T_1, T_2, \ldots, T_n\}$ [15]. This systemic multidimensionality enables topological reasoning that is highly resistant to data poisoning [10].

C. Collaborative Frameworks and Token Efficiency

Recent studies on LLM-Embedding Collaborative Frameworks (e.g., LEC-KG) demonstrate that topological perception significantly enhances domain-specific graph construction [16], [17]. However, relying on cloud-based APIs introduces latency incompatible with autonomous Proto-AGI loops.

Deploying local LLMs (via Ollama or vLLM) solves privacy and latency issues but introduces severe context constraints [18]. To maintain the real-time interaction between the inference engine and the n-ary hyperontology, token efficiency is critical. Benchmarks comparing serialization formats (JSON, YAML, TOON) reveal that standard JSON incurs significant token overhead [19]. This overhead creates a bottleneck for real-time inference, necessitating a more streamlined approach to data representation. By selecting hyper-optimized data structures for our extraction pipelines, the system drastically reduces context bloat, enabling the local MoE model to process expansive topological contexts without memory overflow.

III. System Architecture and Visualization

A. Hyperontology Topological Reification

The core of the Proto-AGI architecture relies on a persistent mathematical hyperontology, which demands significant computational resources for real-time visualization and interaction. To address the inherent complexity of heterogeneous structuring in ontology design, we implemented a dynamic subgraph extraction engine.

Traditional knowledge graphs utilize simple binary edges, but hyperontologies require n-ary relations to accurately represent complex logical pluralism and domain-specific semantic frameworks. To render these multidimensional hyperedges within standard WebGL environments (such as 3D Force-Directed Graphs), our system employs a topological reification technique. Each conceptual hyperedge is dynamically transformed into an intermediate relational node, denoted as R_k.

ISSN 2395-8618

Let $G = (V, E)$ be the local subgraph extracted from the global hyperontology. For each hyperedge $e \in E$ connecting a source node v_s and a target node v_t, the system generates a unique identifier and injects R_k such that the new standard edges become (v_s, R_k) and (R_k, v_t). This architectural decision allows the visualizer to represent logical relations as distinct interactive polygons, facilitating navigation without compromising the mathematical integrity of the underlying hyperontology.

B. Local Subgraph Extraction and Memory Optimization

To ensure computational optimization in local inference engines and prevent memory overflow in the client environment, the engine calculates the connectivity degree (D_v) for each node v. The local extraction algorithm enforces a strict rendering threshold for the neighborhood exploration ($10 \leq N_{limit} \leq 300$).

This selective bounding of the graph topology ensures consistent topological robustness and prevents client-side rendering collapse. By isolating the semantic tree of the central node (the semantic hub) from the global 20,000-node structure, the system guarantees high frame rates during interactive topological reasoning.

IV. Methodology and Implementation

To operationalize the theoretical principles of the Proto-AGI architecture, we developed a local, highly optimized pipeline capable of extracting unstructured data and compiling it into an n-ary hyperontology. The implementation relies on a sequential modular architecture deployed on a local Linux environment, utilizing localized LLMs to ensure delay-robust topological reasoning.

A. The Semantic Extraction Pipeline

The construction of the hyperontology is orchestrated through a multi-stage Python pipeline that completely bypasses external cloud dependencies. The workflow is divided into three primary components:

1) Unstructured Data Parsing and Semantic Partitioning: The initial phase utilizes a customized extraction module to process raw textual inputs, specifically targeting domain corpora (e.g., indigenous language texts such as Náhuatl). To preserve semantic boundaries while strictly adhering to the context window limits of the localized LLM, the text must be partitioned into contextually dense, token-bounded chunks.

Let D represent the raw unstructured document. The system first applies a cleansing function to remove syntactic noise, yielding a normalized text string. A natural language tokenizer function, τ_{sent}, partitions this text into a finite, ordered sequence of sentences, $S = (s_1, s_2, \ldots, s_N)$.

To guarantee that no chunk exceeds the inference engine's memory constraints, we define a tokenization function $T(s_i)$ which returns the exact token length of a sentence s_i using a byte-pair encoding (BPE) algorithm (specifically, `cl100k_base`).

The objective is to partition the sequence S into a set of contiguous ordered sub-sequences (chunks) $C = \{C_1, C_2, \ldots, C_M\}$. The extraction module implements a greedy partitioning algorithm where each chunk C_k is defined as a sequence of consecutive sentences $C_k = (s_i, s_{i+1}, \ldots, s_j)$ subject to the capacity constraint:

$$\sum_{x=i}^{j} T(s_x) \leq \theta_{max}.$$

where θ_{max} is a predefined hyperparameter representing the maximum token threshold per chunk (experimentally set to 3000). The algorithm iterates through S; if the inclusion of s_{j+1} violates the inequality such that $\sum_{x=i}^{j+1} T(s_x) > \theta_{max}$, the current subset C_k is finalized, and a new subset C_{k+1} is initialized starting with s_{j+1}.

Finally, the ordered set of chunks C is serialized into a lightweight JSON array. This output serves as the highly controlled, token-safe input payload for the subsequent semantic assembly engine.

2) Local LLM Semantic Assembly: Once the data is chunked, it is passed to the semantic assembler (`semantic_assembler.py`). This is the core cognitive loop of the framework. Instead of asking the local LLM to generate flat JSON objects, the assembler uses highly constrained prompt templates to enforce the extraction of multidimensional relations. The LLM identifies central concepts (nodes) and their systemic interactions (hyperedges). Because this process runs locally, the system can iteratively query the LLM to resolve logical ambiguities without incurring cloud latency or API token costs.

3) Hyperontology Compilation: The extracted entities and relations are inherently fragmented. The assembly module (`assemble_protoagi.py`) acts as the topological weaver. It receives the disjointed n-ary relations from the semantic assembler and mathematically integrates them into the global graph structure. During this phase, the system resolves node duplications, calculates the connectivity degree (D_v) for each hub, and establishes the strict boundaries required to prevent logical poisoning, as illustrated in the processing pipeline (Fig. 1).

B. Data Serialization, Reification, and Visualization

To bridge the Python-based cognitive backend with the WebGL visualization frontend, the data must be serialized. However, to maximize efficiency and respect the memory constraints of the client browser, we bypass standard, token-heavy JSON generation during the internal LLM loops.

A dedicated conversion script (`convert.py`) translates the compiled Python dictionaries into a hyper-optimized JSON schema specifically designed for the visualizer. It is during this final serialization step that the topological reification—converting abstract hyperedges into physical

ISSN 2395-8618

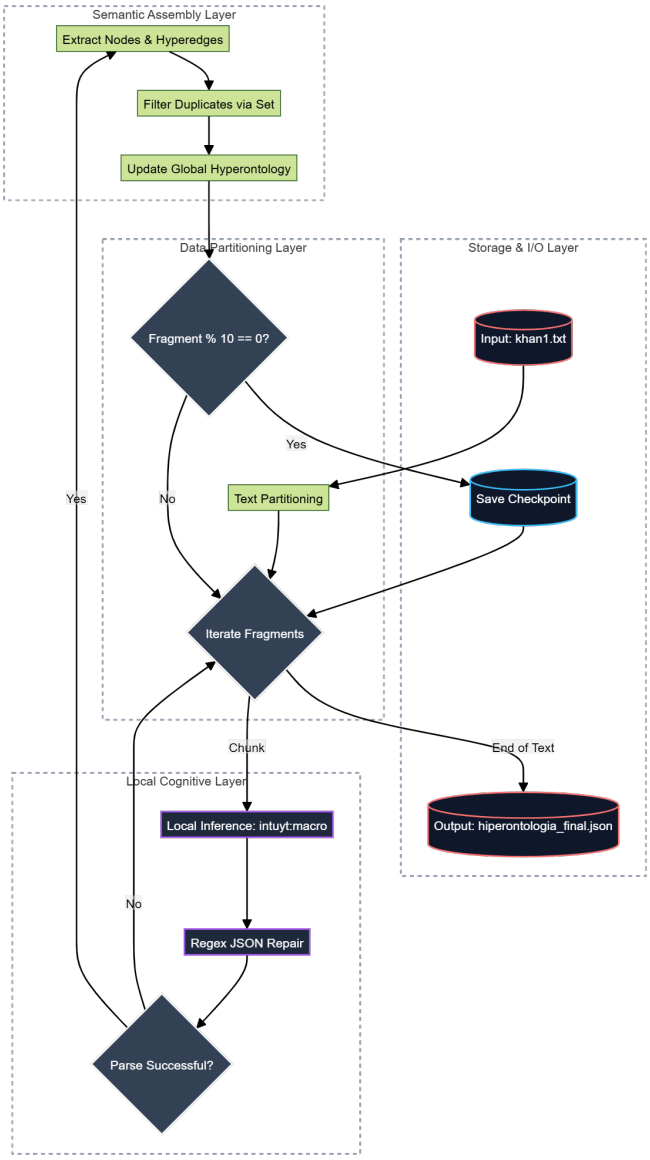

Fig. 1. Systemic architecture of the iterative knowledge extraction and semantic assembly pipeline. The workflow is partitioned into four modular layers: Storage & I/O for data persistence, Data Partitioning for text chunking, Local Cognitive Layer for LLM inference and JSON sanitization, and the Semantic Assembly Layer for global hyperontology integration and duplicate filtering using Set Theory.

intermediate relational polygons—is hardcoded into the output file (`hiperontologia_final.json`).

The resulting serialized data is directly consumed by an interactive WebGL-based visualizer hosted at https://trai-l.com/hyperonto/reader-math.php. This PHP-based frontend (`reader-math.php`) handles the dynamic rendering of the n-ary hypergraph. It employs a selective rendering algorithm, controlled by a user-defined threshold (N_{limit}), to extract and visualize localized subgraphs around a central semantic hub. This ensures that the frontend only performs rendering tasks on manageable topologies, leaving all computational heavy lifting to the backend pipeline while providing an interactive and fluid experience for the end-user.

V. Computational Optimization and Experiments

The deployment of the Proto-AGI architecture in a local environment necessitates severe computational trade-offs. To validate our methodology, we evaluated the system across two primary bottlenecks: inference efficiency during the extraction phase, and client-side rendering capabilities during the visualization phase.

A. Local Inference Efficiency and Domain Specialization

A core tenet of our architecture is the elimination of cloud-based APIs to ensure delay-robust, private, and continuous cognitive loops. To achieve this, the semantic extraction pipeline is powered by `intuyt:macro`, a custom fine-tuned Mixture of Experts (MoE) large language model based on the Qwen3 architecture.

While the model maintains a high total parameter count to preserve broad linguistic and structural knowledge, the MoE architecture routes tokens such that only 3 billion active parameters are utilized during any single inference step. Although our primary research and compilation environment relies on high-throughput enterprise hardware (AMD Instinct MI210 accelerators with 128GB VRAM and 384GB system RAM), the fundamental objective of this active-parameter constraint is accessibility. By restricting the active compute footprint, the semantic extraction pipeline can execute entirely on standard, consumer-grade hardware, adhering strictly to GreenAGI principles of energy efficiency and hardware democratization.

Furthermore, `intuyt:macro` was explicitly fine-tuned on specialized corpora encompassing Mexican Spanish, indigenous languages (such as Náhuatl), and targeted hyperontology structural queries. This domain-specific alignment yields three distinct experimental advantages over massive cloud models:

1) **Token Cost Elimination:** The iterative nature of the greedy partitioning algorithm demands thousands of LLM calls. Local inference reduces the financial cost to pure electricity overhead.
2) **Structural Adherence:** The fine-tuning on hyperontology queries drastically improves the deterministic success rate of the JSON repair heuristic ($\mathcal{H}(R_i)$).
3) **Cultural and Linguistic Fidelity:** Standard cloud models often hallucinate or fail to capture the semantic nuances of low-resource languages, whereas the localized `intuyt:macro` preserves semantic integrity during the topological translation.

B. WebGL Rendering and Topological Bounding

The extraction pipeline easily compiles hyperontologies exceeding 20,000 nodes. However, rendering this global

ISSN 2395-8618

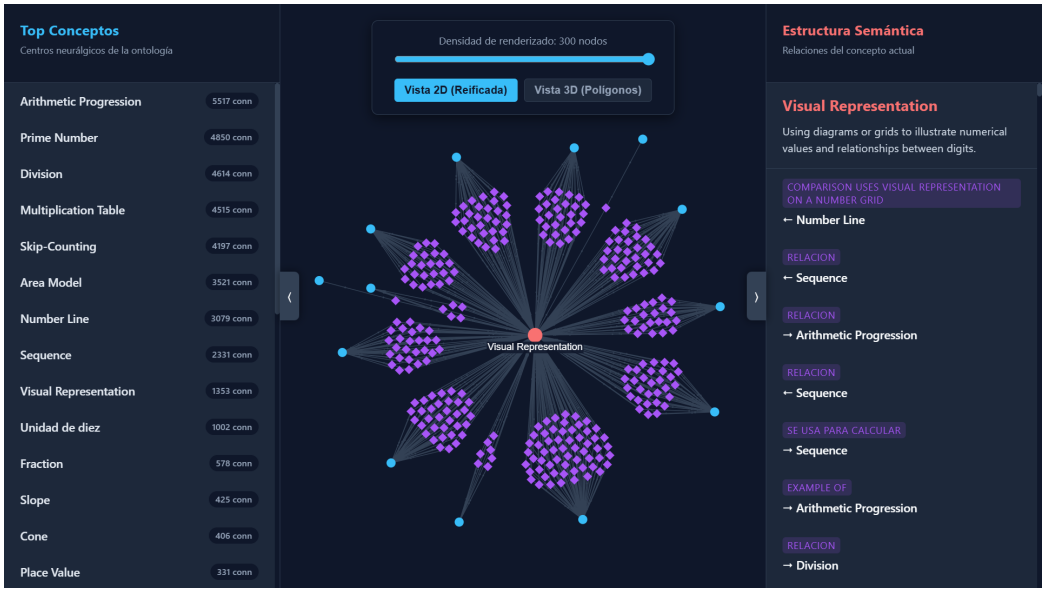

Fig. 2. Dynamic topological reification and local subgraph extraction rendered via the WebGL frontend at trai-l.com. The red sphere denotes the central semantic hub, blue spheres represent adjacent conceptual nodes, and purple tetrahedrons illustrate the reified n-ary relations interconnected by directional particle streams.

graph in a client-side browser introduces critical rendering constraints. The interactive frontend utilizes a 3D Force-Directed Graph engine relying on WebGL and Three.js.

In such physical simulation environments, the computational complexity for calculating repulsive and attractive forces scales quadratically, $O(|V|^2)$, where $|V|$ is the number of rendered nodes. Attempting to render the global hyperontology causes catastrophic frame rate degradation (dropping below 5 FPS) and browser memory overflow.

To resolve this, our experiments empirically established an optimal threshold for the local subgraph extraction ($N_{limit} = 300$). By capping the dynamic rendering to a maximum of 300 conceptually adjacent nodes and reified n-ary relations, the frontend maintains a fluid 60 FPS interaction rate. This topological bounding proves that visualizing a massive Proto-AGI memory structure does not require global rendering; instead, extracting the localized semantic neighborhood around an active conceptual hub is sufficient for effective topological reasoning.

VI. Conclusion and Future Work

This paper formalizes a computationally sustainable architecture for Proto-AGI loops, successfully transitioning the cognitive memory layer from flat Knowledge Graphs to multidimensional mathematical hyperontologies. By synthesizing General Systems Theory with local MoE inference and topological reification, we demonstrated that processing and visualizing semantic networks exceeding 20,000 nodes is entirely viable in edge environments. The integration of greedy token-partitioning and deterministic JSON sanitization effectively prevents logical poisoning while bypassing the latency and privacy bottlenecks inherent to cloud APIs.

Future research will focus on two primary trajectories. First, the linguistic and cultural robustness of the local inference engine will be expanded to encompass broader sets of low-resource indigenous languages, including the automated structural translation of Purépecha. Second, the architecture will transition from a passive knowledge compiler into an active, tool-using Proto-AGI loop. By leveraging the n-ary topological reasoning established in the global hyperontology, the system will be equipped to execute autonomous, multi-step actions, bringing us closer to a persistent, self-governing cognitive architecture.

Code and Data Availability

To ensure the reproducibility of the methodology and foster open-source contributions to Proto-AGI architectures, the complete codebase has been made publicly available. The repository includes the Python semantic extraction pipeline, the local LLM inference configurations for `intuyt:macro`, and the PHP/WebGL frontend for topological visualization.

The source code and the generated n-ary hyperontology datasets can be accessed via the official GitHub repository at https://github.com/christianemaldonadomti/Proto-AGI-Hyperontology.
Furthermore, the interactive WebGL visualization environment remains publicly hosted and operational for peer review at https://trai-l.com/hyperonto/reader-math.php.

ISSN 2395-8618

ACKNOWLEDGMENTS

The authors acknowledge the investigative frameworks of ArtInt.Tech and the open science visualization paradigms of trai-l.com. This research was supported by the generous infrastructure donation, assembly, and hosting provided by DELL, SummaIT and IPICYT. Specifically, the compute environment comprising dual AMD Instinct MI210 accelerators, 384 GB of RAM, and dual EPYC processors was made available through this collaboration. The authors affirm that IPICYT, DELL, and SummaIT were not involved in the design, execution, or analysis of the experiments presented in this work, and no Conflicts of Interest (CoI) exist regarding this hardware contribution. Finally, the author acknowledges the use of advanced generative AI models as a rapid prototyping (Vibecoding) assistant for LaTeX syntax generation, data serialization formatting, and manuscript structuring. The architectural design, theoretical grounding, and systemic logic remain the sole intellectual contribution of the author.

REFERENCES

[1] C. E. Maldonado-Sifuentes, M. Vargas-Santiago, S. Solis-Gamboa, G. Sidorov, L. Lechuga-Gutierrez, F. González-Andrade, and M. Heras-Sánchez, "Towards a proto artificial general intelligence: The role of large language model ontologies in its development," *Computación y Sistemas*, vol. 28, no. 4, pp. 1401–1415, 2024.

[2] GitHub Repository, "A practical architecture for moving from a local assistant/controller toward a persistent, memory-backed, tool-using proto-agi loop," https://github.com, 2024.

[3] Hugging Face, "Open npc ai: Design principles of a proto-agi society," https://huggingface.co, 2024.

[4] Reddit - r/singularity, "Proto-agi [i.e. zombie agi] as an advanced conversational agent," 2024.

[5] LessWrong, "Artifical expert/expanded narrow intelligence, and proto-agi," 2024.

[6] L. von Bertalanffy, *General System Theory: Foundations, Development, Applications*. New York, NY, USA: George Braziller, 1968.

[7] arXiv preprint, "Integrating graphs, large language models, and agents: Reasoning and retrieval," *arXiv preprint*, 2024.

[8] J. O. Franco Franco, "Caminatas aleatorias en hipergrafos y en hiperredes: Los efectos de procesos de reinicio estocástico," Tesis de Licenciatura, Universidad Nacional Autónoma de México, Ciudad Universitaria, Cd. Mx., 2025.

[9] X. Ouvrard, "Hypergraphs: An introduction and review," CERN, Tech. Rep., 2020. [Online]. Available: https://cds.cern.ch/record/2722131

[10] arXiv preprint, "Logicpoison: Logical attacks on graph retrieval-augmented generation," *arXiv preprint*, 2024.

[11] ResearchGate / BIA, "Carnap, goguen, and the hyperontologies: Logical pluralism and heterogeneous structuring in ontology design," 2024.

[12] C. E. Maldonado-Sifuentes, "Proto artificial general intelligence: Proof of concept and proposed architecture," *Polibits*, 2024, preprint.

[13] C. E. Maldonado-Sifuentes, G. Sidorov, O. Kolesnikova, and C. Caballero, "Knowledge and context compression via question generation," *Research in Computing Science*, 2020.

[14] ResearchGate, "The distributed ontology language (dol): Ontology integration and interoperability," 2024.

[15] CEUR-WS.org, "Hyperontology for the biomedical ontologist: A sketch and some examples," 2024.

[16] arXiv preprint, "Lec-kg: A llm-embedding collaborative framework for domain-specific knowledge graph construction — a case study on sdgs," *arXiv preprint*, 2024.

[17] PMC, "Phygeo-kg: Physics-regularized distant supervision for multimodal geometric knowledge graph construction in catenary maintenance," 2024.

[18] Seypro, "Deploying local llms for enterprise: Ollama, vllm, and rag pipelines," 2026.

[19] Zenn, "Json vs yaml vs toon: Comparing token efficiency for llms," 2024.