

Concepts in Context:

The Large Concepts Model (LCM) and the Future of Neuro-Symbolic AI

Wayne Parham, Independent Researcher

wayne@parhamdata.com

Abstract

While modern AI has successfully entrenched itself across the workplace as a general-purpose assistant, its deployment in high-stakes domains – such as scientific analysis, engineering, medical research, and financial planning – is continually undermined by a lack of deterministic accuracy and structural analogical reasoning. As organizations push beyond the limits of probabilistic approximations and mandatory error disclaimers, the demand for rigorous, verifiable intelligence is paramount.

Recent industry efforts have made significant strides toward these goals by introducing agentic workflows and stateful "System 1 / System 2" logic. These architectures excel at iterative, test-break-fix learning loops, functioning somewhat like cellular automata to empirically evaluate and restructure attempts until a solution is found. However, while these approaches improve multi-step execution, they neglect a crucial pillar of advanced cognition: analogical reasoning. Establishing fluid analogies was one of the foundational goals of early AI research, yet it remains profoundly undeveloped in modern applications.

A robust cognitive system must possess the ability to apply structural analogies to navigate novel problems based on verified precedents – for example, recognizing that the harmonic resonance of a pendulum structurally mirrors a mass-spring system or an electronic resonator, and using that isomorphism to theorize the mechanics of ultracold molecular resonances, resonance-assisted quantum tunneling, or biological vibrational resonances.

To bridge this gap, we propose the Large Concepts Model (LCM): a deterministic, neuro-symbolic architecture heavily guided by the active-symbol and mental-fluidity frameworks of Douglas Hofstadter and Melanie Mitchell, the stateful world-model paradigm of Yann LeCun, and the dual-process (System 1 / System 2) cognitive orchestration described by Demis Hassabis.

The LCM strictly grounds the linguistic fluency of Large Language Models (LLMs) by operating on a decentralized "Library and Swarm of Librarians" paradigm rather than treating the LLM as a monolithic reasoning engine. An autonomous, self-healing concept graph acts as a highly structured, deterministic Library housing verified facts, structural ontologies, and executable mathematics. LLMs are operationalized strictly as a specialized swarm of librarians that curate, catalog, and navigate this knowledge. By routing decision-making through a cognitive state machine and multi-dimensional feature vectors – and by tracking emergent conceptual co-occurrences – the LCM transcends the brittle heuristics of standard Retrieval-Augmented Generation (RAG). It provides not only the absolute accuracy required for zero-tolerance environments but also the verifiable analogical reasoning necessary to drive genuine scientific and engineering discovery.

1. Introduction and Foundations

The contemporary AI paradigm is constrained by a reliance on high-dimensional semantic spaces that map statistical correlations between lexical tokens rather than conceptual structures. While Large Language Models demonstrate linguistic fluency, they lack an underlying cognitive architecture capable of genuine abstraction.

This limitation reflects foundational critiques in cognitive science. Douglas Hofstadter's work on the Copycat architecture demonstrated that fluid analogical thought requires structured, slippable conceptual networks rather than brute-force statistical search. Hofstadter established that human abstraction depends on "conceptual slippage" – the capacity of concept representations to stretch and align under structural constraints. Melanie Mitchell formalized this challenge through benchmarks such as ConceptARC, proving that autoregressive transformers degrade on relational and geometric analogies because they lack explicit spatial rules. Furthermore, recent evaluations such as the ANALOGICAL benchmark, championed by Amit Sheth and colleagues, have empirically proven that as textual and structural complexity scales, LLMs inevitably abandon true relational mapping in favor of superficial lexical matching.

Concurrently, Yann LeCun has argued for autonomous "world models" through frameworks like Joint Embedding Predictive Architectures (JEPA), emphasizing the necessity of separating probabilistic language prediction from persistent, stateful models of reality. Demis Hassabis has highlighted the need to bridge intuitive pattern recognition (System 1) with deliberate, algorithmic reasoning (System 2). Furthermore, while LLMs demonstrate proficiency in semantic generation, they inherently fail at strict multi-step reasoning and constraint-aware planning – a limitation recently formalized by Sheth et al. (2025), necessitating a transition toward neuro-symbolic knowledge grounding.

Current enterprise architectures fail to resolve these challenges due to three distinct bottlenecks:

- **Pure LLMs:** Knowledge remains trapped in static neural weights. Fact updates require cost-prohibitive fine-tuning, and mathematical execution relies on syntactical token prediction.
- **Retrieval-Augmented Generation (RAG):** Standard RAG functions as a surface-level text-matching heuristic, forcing the LLM to parse and resolve unstructured, unverified context dynamically during generation. As noted in recent neuro-symbolic literature (Soni & Sheth, 2026), non-parametric augmentation like RAG fails to deterministically restrict an LLM from misinterpreting retrieved context.
- **Unconstrained Agentic Frameworks:** Exposing an LLM to a global tool registry overwhelms token selection logic, inducing tool confusion, inconsistent parameter binding, and hallucinated payloads.

To resolve these failure modes, the Large Concepts Model (LCM) enforces a strict neuro-symbolic boundary between the cognitive agents navigating the data and the underlying concept repository.

2. Architectural Paradigm: The Library and the Swarm

Building upon the imperative to elevate LLM operations from token-level statistical matching to higher-order conceptual abstractions (Wijesiriwardene, Thirunarayanan, & Sheth), the Large Concepts Model implements a DAG-based architecture to enforce strict analogical boundaries. The LCM replaces ungrounded text retrieval with atomic, multi-dimensional Concept Definitions. The system models knowledge not as flat documents or vector embeddings, but as a Directed Acyclic Graph (DAG) that serves as an authoritative Library. The LLMs operate externally as an asynchronous Swarm of Librarians tasked with cataloging and serving this library.

2.1 System 1 and System 2 Division of Labor

During runtime inference, a lightweight LLM functions as a System 1 "Reference Librarian." It parses conversational intent and matches incoming queries to indexed concept nodes. When a task requires deterministic calculations, formal logic, or multi-step execution, the Reference Librarian delegates the operation to System 2 "Concept-Bound Agents." These agents execute deterministic code (e.g., Python scripts, REST or MCP calls, complex mathematical expression evaluators or other symbolic solvers) bound directly to the relevant nodes, returning exact outputs.

2.2 Anatomy of a Concept Node

Every entity, mathematical structure, or operational pattern within the LCM is represented as an isolated, schema-enforced concept node containing:

- **Ontology (isA):** Vertical inheritance pointers linking the node to broader abstract parents within the DAG.
- **Lexical Labels:** Curated synonyms, aliases, and keyword triggers.
- **Atomic Observations:** A verified collection of discrete, testable facts, properties, and constraints.
- **Mathematical & Scientific Formulas:** Encoded, normalized equations (e.g., AST-mapped representations of physical laws or geometric properties) that allow the system to execute computations and detect isomorphic relationships across domains.
- **Algorithmic & Computational Paradigms:** Explicit representations of software patterns, recognized algorithms, essential variables, methods, and systemic behaviors. These computational signatures are integrated directly into the composite feature vector, enabling the system to recognize structural logic and operational equivalencies.
- **Concept-Bound Agents:** Isolated computational scripts (e.g., Python) or declarative API/MCP definitions attached strictly to the parent concept.

2.3 The Cognitive Orchestrator and Stateful Execution

Without a state engine, an architecture is merely a collection of models associated by vectors. The LCM introduces a Cognitive Orchestrator equipped with an Execution Scratchpad (short-term memory) to process recursive and multi-step tasks. Instead of statically predicting a sequence of operations upfront, the Orchestrator runs a continuous evaluation loop:

1. **Analyze & Retrieve:** The system evaluates the current state of the problem and calls upon specific concepts and their bound agents dynamically.
2. **Act & Mutate:** The deterministic agent (e.g., a native Breadth-First Search graph traversal) executes and returns a result.
3. **Observe & Critique:** The Orchestrator logs the result in its state memory, re-examines the shifting physical boundaries or variables, and determines the next logical step.

2.4 Localized Tool Routing

By binding agents to discrete concept nodes, the LCM eliminates global tool selection overhead. A query involving polyhedral metrics activates only the relevant geometric nodes (e.g., `REGULAR_ICOSAHEDRON`), exposing only the agents scoped to those nodes. The LLM is structurally constrained to incorporate deterministic outputs rather than generating calculated numbers probabilistically.

3. The Feature Similarity Engine: A New Categorization Paradigm

Standard embedding models evaluate words based on distributional similarity in a shared vector space. Consequently, distinct semantic meanings of identical words – such as "sphere of influence" versus a "geometric sphere" – often cluster together incorrectly due to shared lexical contexts.

The LCM introduces a whole new way of categorizing information and testing for matches. It resolves semantic conflation by pairing a high-dimensional dense embedding (**Essence Vector**) with a multi-axis structural evaluation (**Feature Vector**) and discrete lexical classifiers. The Essence Vector acts as a fast, intuitive filter, while the Feature Similarity Engine performs rigorous topological and algorithmic comparison.

The composite structural score, $S_{feature}(A, B)$, is calculated as:

$$S_{feature}(A, B) = w_{ont}S_{ont} + w_{lab}S_{lab} + w_{form}S_{form} + w_{obs}S_{obs}$$

Where each component isolates a specific axis of conceptual alignment:

- **S_{ont} (Ontological Similarity):** The taxonomic proximity of the concepts within the DAG.
- **S_{lab} (Lexical Similarity):** The overlap of curated synonyms, aliases, and keyword triggers.
- **S_{form} (Formulaic Isomorphism):** The structural equivalence of underlying mathematical, procedural or algorithmic Abstract Syntax Trees (AST).
- **S_{obs} (Observational Overlap):** The semantic intersection of discrete, verified facts and constraints.

3.1 Lexical and Observational Overlap

To prevent high-frequency conversational stop words from skewing observational similarity, unstructured observation text is canonicalized, stripped of stopwords, and evaluated via Jaccard index scoring:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

For structured categorical arrays, such as explicit label sets, the LCM applies the Overlap Coefficient to evaluate subsets without penalizing narrow, highly specific child nodes:

$$O(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

3.2 Ontological Distance Scoring

The ontological similarity function $O(A, B)$ calculates taxonomic proximity within the DAG using the explicit inheritance pointers $P(x)$ of each concept:

$$O(A, B) = \begin{cases} 1.0 & \text{if } A \in P(B) \vee B \in P(A) \\ \frac{|P(A) \cap P(B)|}{\min(|P(A)|, |P(B)|)} & \text{if } P(A) \neq \emptyset \wedge P(B) \neq \emptyset \\ 0.0 & \text{otherwise} \end{cases}$$

If an embedding model conflates SPHERE_INFLUENCE and SPHERE_GEOMETRY, the LCM sets $O(A, B) = 0.0$, because their parent sets share no common ancestor in the DAG. This ensures strict boundary separation across different domains of knowledge.

3.3 Mathematical and Algorithmic Isomorphism

The LCM evaluates functional and algorithmic similarity using canonical Abstract Syntax Tree (AST) normalization. Mathematical formulas and agent code are transformed via a mapping pipeline $N(F)$ that:

1. Replaces arbitrary variable identifiers with standardized indexed placeholders (e.g., $x, y \rightarrow var1, var2$).
2. Reorders commutative operations alphabetically and hierarchically.

For instance, the equations $V = \frac{4}{3}\pi r^3$ and $\text{volume} = (4/3) * \text{PI} * (\text{radius}^3)$ are normalized into identical tree structures. The mathematical similarity score $M(A, B)$ is computed across the normalized AST sets using the Jaccard index (J) defined in section 3.1:

$$M(A, B) = J(N(F_A), N(F_B))$$

This mechanism allows the system to recognize geometric symmetries and dimensional transformations algebraically, enabling structural analogy mapping.

3.4 System Implementation and Environment

To validate the architecture, a working prototype was implemented using the following reference components:

- **Inference Orchestrator:** The Reference Librarian role is not bound to a specific provider. For baseline testing, frontier models (GPT-4o) were utilized via API integration, strictly configured at a temperature of 0.0 to enforce high determinism during schema parsing.
- **Dense Embeddings:** The Essence Vector generation was implemented using standard high-dimensional vector models (text-embedding-3-large producing 1536-dimensional vectors), though any dense semantic embedding model can be substituted.
- **Library Store:** To support scaling from independent research to enterprise deployments, the library abstracts its data persistence. The reference prototype utilizes lightweight filesystem serialization for instantaneous local graph loading, while the enterprise architecture is designed to map the Directed Acyclic Graph onto transactional relational or graph databases (e.g., PostgreSQL).
- **Subconscious Worker Queue:** The background librarian swarm was engineered as decoupled, asynchronous tasks managed via Spring Boot execution schedulers, decoupling graph optimization from runtime query evaluation.

4. The Autonomous "Subconscious": Self-Healing and Transfer Learning

Traditional Knowledge Graphs (e.g., Semantic Web, Neo4j) are inherently brittle because they require human data engineers to define schemas and reconcile edges manually. The LCM introduces an asynchronous, autonomous worker swarm – acting as a tireless staff of background librarians – that maintains graph integrity without human intervention.

Crucially, this continuous curation process represents a novel form of transfer learning. The broad, probabilistic knowledge suspended in the massive latent weights of frontier LLMs is systematically distilled (or "congealed") into discrete, high-fidelity graph nodes within the LCM. By shifting the LLMs away from generation and instead issuing them highly atomic, single-task instructions – a constraint under which LLMs exhibit maximal reliability – the system translates generalized probability into structural certainty.

When the LCM detects a gap in its understanding (such as an unmet inference constraint or a novel query), it initiates an exploration and maturation lifecycle:

- **Exploration and Enrichment (The Atomic Workers):** The swarm scans the incubator for unknown or sparse concepts. Workers are dispatched with intensely focused, single-task prompts to interrogate the LLM. They extract specific structural facets: ontology (isA relationships), empirical observations (facts), procedural steps, software patterns, and underlying mathematical or physical formulas.
- **Subcategorization:** Workers detect overly broad or ambiguous concepts (e.g., COLLECTION) and automatically split them into distinct, specialized senses (e.g., MATH_COLLECTION, PROGRAMMING_COLLECTION), dynamically reparenting orphaned semantic children.
- **Verification and Vetting:** Before any concept is committed to the active library, an independent set of Verification Workers acts as a strict editorial board. They test the newly distilled concepts for factual accuracy, ontological consistency, and logical integrity, assigning the concept a cumulative confidence score.
- **Promotion and Vectorization:** Once a concept achieves a high enough confidence threshold through the vetting process, it is marked for promotion. It is permanently placed into the global parameter list and its narrative essence is vectorized. Upon vectorization, its dimensional axes are mapped, immediately allowing the system to form analogical relationships and structural links with all other known concepts in the library.
- **Cleanup (Garbage Collection):** Concepts that suffer from semantic conflation, internal contradiction, or hyper-fragmentation naturally decay in confidence and are autonomously purged from the system.

5. Emergent Reasoning and Pattern Discovery

Because the LCM maps knowledge deterministically, it is capable of emergent discovery. As the system tracks user queries, the Emergent Concept Worker monitors co-occurrence frequencies between activated concept IDs. When two disparate concepts are frequently invoked together, the system autonomously spawns a new composite concept, researching and defining the bridge between them.

Similarly, the Pattern Discovery Worker identifies structural labels and autonomously generates robust Regular Expression classifiers, moving the system beyond brittle keyword matching without spending expensive LLM tokens during runtime inference.

6. Methodology and Empirical Evaluation

As frontier LLMs saturate static benchmarks via memorized corpora and tackle mathematical evaluations through hidden code execution, benchmark testing in these domains ceases to be a meaningful discriminator of cognitive architecture. While iterative state machines and Python agents excel at solving multi-step arithmetic – and even performing step-by-step code revisions with unit testing – they frequently miss critical details and introduce errors because they fundamentally do not understand the underlying logic of the algorithms they manipulate. The true test of a neuro-symbolic framework lies in its ability to perform fluid structural abstraction, analogical mapping, and formulaic identification – domains where purely probabilistic token prediction and ungrounded code execution prove insufficient.

Test 1: Structural Analogical Mapping

To validate the efficacy of the Feature Similarity Engine in executing structural reasoning and conceptual slippage – without relying on mere lexical proximity – the LCM was subjected to the classic geometric proportional analogy: *“Sphere is to circle as cube is to what?”*

- **Lexical and Entity Identification:** The system classifiers identified the input entities: SPHERE, CIRCLE, and CUBE.
- **Candidate Shortlisting:** The essence vector similarities included a candidate pool of nearest neighbors, which included SQUARE, BOX, SPHERE_GEOMETRY, SOCIAL_SPHERE, and PUBLIC_PRIVATE_SPHERE.
- **Topological Filtering:** Candidates suffering from semantic conflation, such as SOCIAL_SPHERE and PUBLIC_PRIVATE_SPHERE, returned high essence vector similarities due to lexical overlap but were successfully rejected because their composite Feature Scores were suppressed by an $O(A, B)$ of 0.0. Candidates within the correct geometric ontology (such as SPHERE_GEOMETRY or BOX) proceeded to the structural evaluation.

- **Feature Vector Synthesis:** The Feature Similarity Engine independently evaluated the distance between the source pair (SPHERE→CIRCLE) and the target candidates (CUBE→ x), using the following steps:
 - **AST Transformation Mapping:** The system retrieved the bound mathematical definitions of SPHERE ($V = \frac{4}{3}\pi r^3$) and CIRCLE ($A = \pi r^2$), normalizing them into Abstract Syntax Trees to isolate the underlying algebraic transformation (a dimensional reduction from 3D to 2D).
 - **Analogical Resolution:** Applying this operational transformation to the CUBE node identified SQUARE as an exact structural match, yielding an AST isomorphism score of $M(A, B) = 1.0$. The system solved the analogy algebraically via explicit mapping rather than probabilistic next-token prediction.

Test 2: Formulaic Concept Identification (Reverse-Mapping)

To evaluate the system's ability to transcend lexical keyword matching, the LCM was tasked with identifying a concept purely by a raw mathematical signature: *"What things are associated with the formula πr^2 ?"*

Because the query lacks explicit geometric lexical triggers (e.g., "circle," "area," "geometry"), standard semantic vector retrieval often struggles or returns superficial text matches. However, the LCM successfully processed the input, normalized the equation via the Feature Similarity Engine, and matched the mathematical signature directly to the bound properties of the CIRCLE_FILLED_REGION and EUCLIDEAN_CIRCLE_LOCUS nodes. The Cognitive Orchestrator successfully retrieved a cluster of 22 structurally related concepts—including underlying R^2 coordinate plane paradigms—and structurally constrained the System 1 Reference Librarian to output a deterministic, contextually accurate definition of circular area without probabilistic hallucination.

Test 3: Taxonomic Disambiguation and Anti-Conflation

To evaluate the system's resilience against statistical semantic conflation, the LCM analyzed two highly correlated nodes to determine if they should be automatically merged: SPHERE and SPHERE_GEOMETRY.

- **Semantic and Mathematical Overlap (The RAG Trap):** Standard dense vector embeddings yielded a high cosine similarity of 0.811 due to shared domain vocabulary. Furthermore, the Feature Similarity Engine confirmed an exact mathematical isomorphism score of $M(A, B) = 1.0$, as both nodes share identical bound formulas (e.g., $V = \frac{4}{3}\pi r^3$ and $A = \pi r^2$). A standard ungrounded LLM or pure vector database would conflate the two, treating them as interchangeable tokens in the same semantic cluster.

- **Ontological Guardrails:** The LCM's decoupled feature engine evaluated the explicit inheritance pointers of both concepts. SPHERE descends from THREE_DIMENSIONAL_SHAPE (a spatial solid), whereas SPHERE_GEOMETRY descends from EUCLIDEAN_GEOMETRY (a branch of mathematics).
- **Resolution:** Because their parent taxonomies diverged completely, the intersection of their parent sets was empty, yielding an ontological similarity score of $O(A, B) = 0.0$. This topological filter acted as a strict categorical guardrail, preventing the system from making the fundamental type error of treating a physical object as an academic discipline. The structural separation enabled the system to maintain pristine graph integrity without falling into the statistical word traps inherent to pure language models.

7. Conclusion

At its core, a pure Large Language Model operates entirely in the subsymbolic realm. While it ingests and emits discrete symbols (text and punctuation), its internal reasoning is a continuous navigation of high-dimensional latent space.

Humans rely heavily on this kind of subsymbolic processing for sensorimotor tasks – such as learning to ride a bicycle. We do not consciously calculate the exact torque or angular momentum required to maintain balance while accelerating or turning; we simply adjust the weights and biases of our neural pathways through repeated practice. We often call this "muscle memory," but it is fundamentally a subsymbolic training process. Early human language acquisition follows a similar pattern: babies begin with probabilistic phonetic mimicry, learning to sequence sounds before they understand meaning.

However, true human cognition emerges when this subsymbolic fluency crosses the threshold into symbolic representation. A child eventually learns to map those sounds to discrete concepts. They learn to establish cause-and-effect relationships (e.g., flipping a switch illuminates a room), and critically, they begin to apply analogical reasoning to novel situations based on those established symbols. This is the path to cognition, and it is the goal of the LCM.

Current enterprise AI is largely trapped in the phonetic mimicry and muscle-memory phase. The Large Concepts Model represents the necessary developmental leap. By delegating statistical language processing to a swarm of intelligent LLM librarians and deterministic knowledge mapping to an autonomous, self-healing graph library, the LCM eliminates hallucination, enables true analogical reasoning, and scales enterprise knowledge dynamically. It is not just an alternative to RAG; it is the blueprint for reliable, neuro-symbolic AI.

References

- Hofstadter, D. R., & Mitchell, M. (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books.
- Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.
- LeCun, Y. (2022). *A Path Towards Autonomous Machine Intelligence*. OpenReview. <https://openreview.net/forum?id=BZ5a1r-kVsf>
- Moskvichev, A., Odouard, V. V., & Mitchell, M. (2023). The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain. *arXiv preprint arXiv:2305.07141*.
- Bhavya, B., Jin, D., Huang, H., Luo, M., Hu, X., & Sheth, A. (2023). ANALOGICAL - A Novel Benchmark for Long Text Analogy Evaluation in Large Language Models. *Findings of the Association for Computational Linguistics: ACL 2023*.
- Sheth, A., Khandelwal, V., et al. (2025). NeuroSymbolic Knowledge-Grounded Planning and Reasoning in Artificial Intelligence Systems. IEEE Intelligent Systems.
- Wijesiriwardene, T., Thirunarayanan, K., & Sheth, A. From Morphemes to Knowledge Graphs: Enabling Abstractions in Large Language Models With Neurosymbolic AI. IEEE Intelligent Systems.
- Soni, V., & Sheth, A. (2026). Symbolic-in-the-Loop: Leveraging Knowledge Graph Engineering for Neuro-Symbolic Integration. IEEE.