SEMANTIC TOPOLOGY REASONING ARCHITECTURE (STRA): FROM PARAMETER-CENTRIC MODELS TO STRUCTURE-CENTRIC REASONING

Marcelo Emanuel Paradela Teixeira[1]

[1] Independent Researcher

ABSTRACT

Large language models fuse knowledge and reasoning into billions of inscrutable parameters, trading interpretability for performance. We propose Semantic Topology Reasoning Architecture (STRA), which cleanly separates: (1) knowledge as explicit, inspectable semantic topology; (2) reasoning as meta-operations by smaller models (1-7B parameters) trained on topology navigation; (3) language as output interface, not cognitive substrate. This separation enables transparency (visible reasoning paths), efficiency (targeted computation), correctability (edit knowledge without retraining), and genuine cross-domain reasoning through semantic similarity. STRA integrates five primitives: Activation Arrays (working memory), Causal Signatures (cross-domain analogy), Selection Pressure (reasoning stability), Transform Learning (procedural compression), and Semantic Abacus (skill acquisition). These form a complete architecture for transparent, evolvable reasoning that operates on concepts, not tokens.

KEYWORDS

Semantic reasoning, transparent AI, knowledge representation, activation dynamics, explainable AI

1. INTRODUCTION

The rapid scaling of Large Language Models (LLMs) to hundreds of billions of parameters has revolutionized natural language processing. However, this parameter-centric paradigm creates fundamental trade-offs: knowledge is fused with reasoning, resulting in opaque, inefficient, and brittle systems. Recent advancements in Graph Neural Networks (GNN) and neuro-symbolic architectures suggest a path forward, but lack a unified framework. This paper proposes Semantic Topology Reasoning Architecture (STRA), a foundational paradigm that cleanly separates knowledge from reasoning to address these core limitations.

Large language models achieve remarkable capabilities through next-token prediction, yet fundamentally approximate reasoning through statistical pattern matching over sequences. This creates four critical limitations:

(1) Opacity - knowledge distributes across billions of parameters, making inspection impossible;

(2) Inefficiency - every query recomputes through entire parameter space;

David C. Wyld et al. (Eds): CNSA, CCSEIT, NLAI, SIGML, ICCIOT, AIAP – 2026
pp. 135-152, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160514

(3) Brittleness - correcting errors requires retraining rather than editing;

(4) Parameter Inflation - performance improvements require scaling parameters (70B, 175B, 405B), wrongly fusing "what to know" with "how to reason."

1.1. The Core Insight

Current architectures make a fundamental design error: **knowledge and reasoning are orthogonal capabilities that should not share parameters**. Physics knowledge requires different storage than logical inference mechanisms, yet transformers entangle both in shared weights. STRA separates what was wrongly combined:

- **Knowledge** = Explicit semantic topology (navigable concept graph)
- **Reasoning** = Small model (~1-7B params) trained on meta-reasoning
- **Language** = Output interface, not substrate of cognition

This separation is not novel individually - semantic networks [1-3], cognitive architectures [4-6], and GNN [7,8] explored related concepts. STRA's contribution is **unifying these ideas into a complete, integrated architecture** with five interdependent primitives working synergistically.

Why Tokens Fail: Token sequences encode surface regularities but lack persistent identity, causal role, and structural invariants. STRA replaces token adjacency with concept persistence and relational topology, enabling reasoning over structure rather than syntax. This architectural shift moves from statistical co-occurrence to explicit semantic relationships.

1.2. Contribution

This paper presents STRA as a theoretical framework addressing LLM limitations through:

(1) Complete architectural specification separating knowledge, reasoning, and language;

(2) Five integrated primitives forming a coherent reasoning system;

(3) Transparent, inspectable reasoning paths at every step;

(4) Mechanism for genuine cross-domain reasoning through structural analogy;

(5) Pathway from concept to proof-of-concept implementation.

Theoretical Status: STRA should be understood as a conceptual architecture and research hypothesis without empirical validation. All claims regarding efficiency, transparency, and reasoning quality are hypotheses derived from architectural principles that require experimental verification through proof-of-concept implementation. This work establishes the conceptual foundation and implementation pathway for such validation.

2. BACKGROUND

Approach	Key Mechanism	STRA Adoption	STRA Extension
Semantic Networks [1-3]	Concept nodes + relationship edges	Semantic topology foundation	Add activation dynamics, meta-reasoning
Cognitive Architectures [4-6]	Modular knowledge/reasoning	Separation of concerns	Neural meta-reasoning vs symbolic rules
Graph Neural Networks (GNN) [7,8]	Learning over graphs	Topology navigation	Meta-reasoning on abstract patterns
RAG/GraphRAG [9,10]	Augment LLMs with retrieval	External knowledge access	Reasoning operates directly on topology
Neuro-Symbolic [11]	Combine neural + symbolic	Hybrid approach	Continuous topology vs discrete logic

Table 1: Relation to Prior Work

STRA synthesizes these foundations into a unified architecture where: semantic topology provides knowledge substrate (semantic networks); meta-reasoning model learns navigation patterns (GNN-inspired); five primitives handle distinct cognitive functions (cognitive architecture modularity); reasoning operates on topology directly rather than through language mediation (beyond RAG).

3. ARCHITECTURE OVERVIEW

3.1. Three-Layer Separation

As illustrated in Figure 1, STRA implements a three-layer architecture cleanly separating knowledge, reasoning, and language:

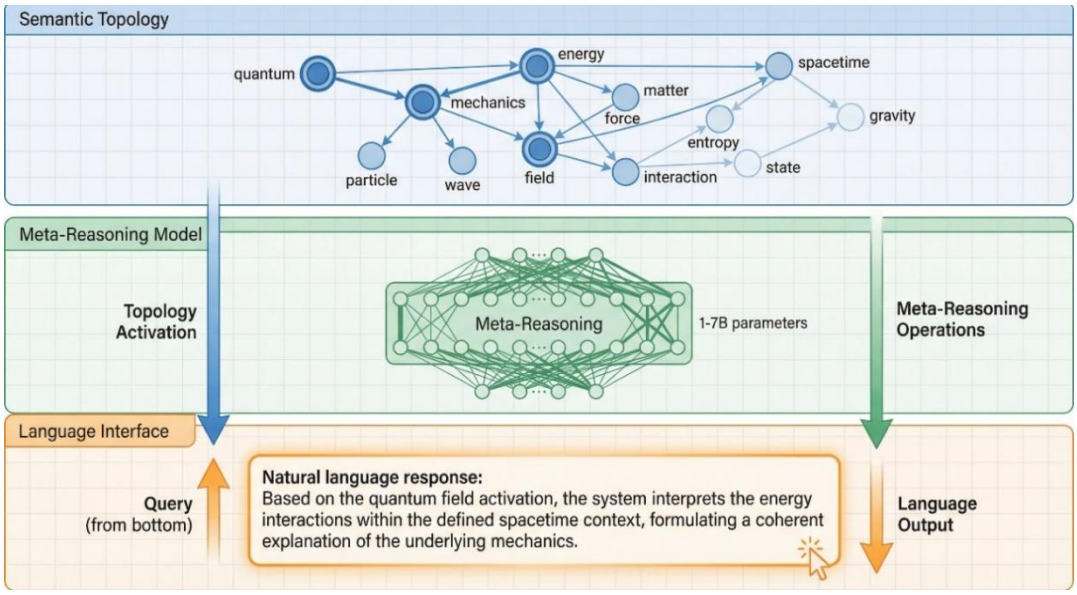

Figure 1. Three-Layer Architecture of STRA showing separation of knowledge (Semantic Topology), reasoning (Meta-Reasoning Model), and language (Language Interface) with information flow from query input through topology activation, meta-reasoning operations, to natural language output.

Layer 1: Semantic Topology (Knowledge) - Directed graph where nodes represent concepts (quantum mechanics, democracy, grandmother), edges represent relationships (is-a, causes,

precedes), and edge weights encode association strength. Human-readable, editable, domain-agnostic.

Layer 2: Meta-Reasoning Model (Thought) - Compact neural network (1-7B parameters) trained on topology navigation patterns, activation propagation dynamics, cross-domain structural mapping, transform composition rules, and quantitative operation sequences. Unlike language models, this model is not trained on text corpora but on abstract reasoning traces over semantic topologies, decoupling linguistic fluency from cognitive competence. This model learns **how to think**, not what to know.

Layer 3: Language Interface (Expression) - Traditional LLM (or smaller specialized model) receiving reasoning results from Layer 2, generating natural language explanations, with no direct access to semantic topology. This model learns **how to express**, not how to reason.

Information Flow: Query → Parse concepts → Activate topology nodes → Meta-reasoning operations → Result structure → Language generation → Response

Comparison to LLMs: While LLMs fuse all three layers into shared parameters, STRA separates them, enabling: transparency (inspect activation patterns), efficiency (compute only relevant topology regions), correctability (edit topology without retraining), compositionality (flexibly combine reasoning primitives).

4. FIVE CORE PRIMITIVES

4.1. Activation Arrays: Semantic Working Memory

Problem: Human reasoning maintains multiple concepts simultaneously. Token-based systems process sequentially, losing parallel context.

Formal Operational Definition:

- **Input:** Query-derived initial concept set, semantic topology subgraph containing activated nodes and their neighbours, previous activation state (if continuing reasoning)
- **Output:** Time-evolved activation vector α(t) where each element represents [concept_id, activation_level∈ [0,1], context_tags, timestamp], maintaining top-k most activated concepts
- **Constraints:** Fixed array capacity k (typically 100-500 slots); activation values normalized per timestep; decay factor δ prevents saturation; lateral inhibition manages competing concepts
- **Mechanism:** Activation Arrays are fixed-size vectors (100-500 slots) where each slot holds [concept_id, activation_level, context_tags, timestamp]. Activation propagates through topology edges:

$$\alpha_{t+1}(v) = \alpha_t(v) \cdot \delta + \Sigma_{u \in N(v)} w(u,v) \cdot \alpha_t(u)$$

where δ is decay factor, N(v) is neighbourhood, w(u,v) is edge weight. Activation values are bounded to [0,1] with normalization applied per timestep to maintain stable dynamics. Hebbian learning [12] strengthens frequently co-activated connections. Figure 2 illustrates the temporal

evolution of activation patterns across the array, showing how concepts spread through the topology and compete for limited working memory slots.

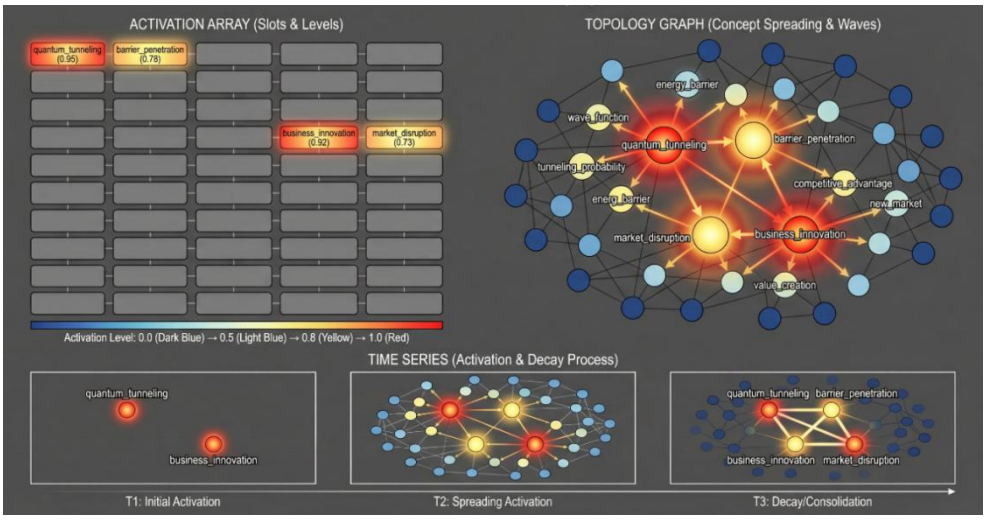

Figure 2. Activation Array dynamics showing temporal evolution of concept activation levels, spreading activation through topology edges, and competitive inhibition among related concepts over multiple timesteps.

- **Operations:** Selective attention (boost query-relevant concepts), decay (prevent saturation), inhibition (suppress contradictions), consolidation (strengthen co-activated connections via Hebbian learning [12]).

- **Working Memory Dynamics:** Meta-reasoning model learns to maintain task-relevant concepts at high activation, prune irrelevant activations, and balance exploration (spread widely) vs exploitation (focus on strong paths). Array capacity management uses threshold gating, competitive inhibition among similar concepts, and context tagging to track activation sources.

Example: Query "How does quantum tunnelling relate to business innovation?" activates both "quantum tunnelling" cluster (barrier penetration, probability, forbidden transitions) and "business innovation" cluster (risk-taking, market disruption, paradigm shift) simultaneously, enabling analogy detection.

4.2. Causal Signatures: Cross-Domain Structural Analogy

Problem: LLMs struggle with genuine cross-domain reasoning - recognizing that water flow, traffic congestion, and electrical current share deep structural patterns beyond surface similarities.

Formal Operational Definition:

- **Input:** Activated subgraph from semantic topology (typically 20-100 nodes and their relationships), domain context identifier, temporal evolution window (if applicable)

- **Output:**Causal Signature $\Sigma = \{V_{roles}, E_{causal}, C_{constraints}, D_{dynamics}\}$ encoding abstract structural patterns; similarity score $sim(\Sigma_A, \Sigma_B) \in [0,1]$ for cross-domain matching

- **Constraints:** Signature extraction preserves structural invariants (feedback loops, bottlenecks, cascades) while discarding surface features; role mapping must be isomorphic; minimum similarity threshold θ (typically 0.75) required for valid analogy

- **Mechanism:** Extract abstract causal structure from domains. A Causal Signature Σ is a graph encoding:

$$\Sigma = \{V_{roles}, E_{causal}, C_{constraints}, D_{dynamics}\}$$

Where V_{roles} are entity types with roles (source, sink, medium, barrier, catalyst), E_{causal} are causal relationships (drives, inhibits, mediates), $C_{constraints}$ are boundary conditions, $D_{dynamics}$ are temporal evolution patterns. As illustrated in Figure 3, this extraction process identifies structural patterns across domains, enabling analogical reasoning through signature matching.

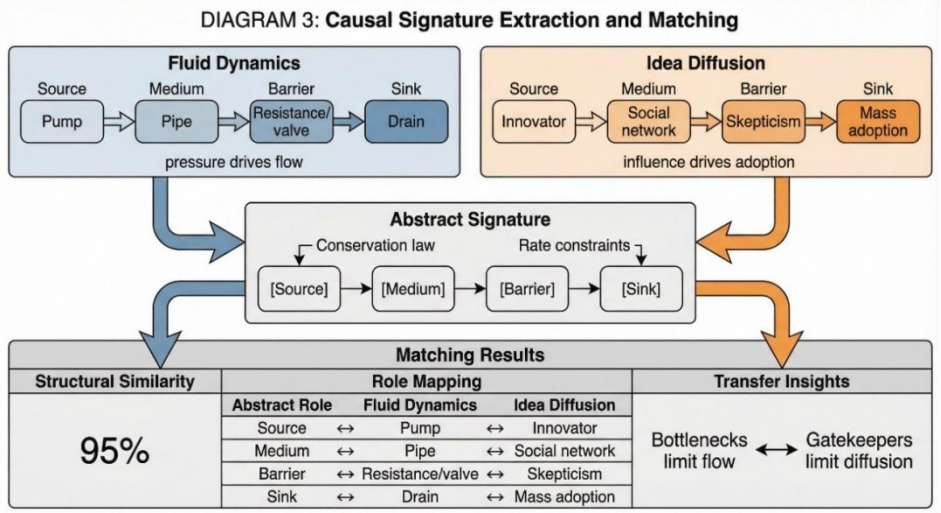

Figure 3. Causal Signature extraction and cross-domain matching demonstrating structural analogy between fluid dynamics and idea diffusion domains through isomorphic role mapping.

Signature Extraction: Meta-reasoning model identifies causal roles, relationship types, structural invariants (feedback loops, bottlenecks, cascades), and scale parameters (linear, exponential, saturating). Two domains are analogous if their signatures are isomorphic under role mapping:

$$similarity(\Sigma_A, \Sigma_B) = 1 - edit\_distance(\Sigma_A, \Sigma_B) / max\_distance$$

Example: Fluid dynamics signature (pump → pipe → resistance → drain) maps to idea diffusion (innovator → network → skepticism → adoption) with 95% structural similarity, enabling insight transfer: "bottleneck in pipes" ↔ "gatekeepers in networks."

4.3. Selection Pressure: Evolutionary Stability of Reasoning

Problem: Without constraints, spreading activation produces spurious connections. Human cognition uses selection mechanisms to stabilize reasoning.

Formal Operational Definition:

- **Input:**Population $P = \{p_1, p_2, \ldots, p_n\}$of candidate reasoning paths (each path is a sequence of topology traversals and transform applications), fitness weights (α, β, γ, δ) for evaluation dimensions, task context specifying selection strength

- **Output:** Filtered population $P' \subseteq P$ containing high-fitness paths (typically top 20-30%), diversity metrics, convergence indicators

- **Constraints:** Population size bounded (10-1000 paths); fitness function normalized; mutation rate $\mu \in$ [0.01, 0.1] maintains exploration; extinction threshold eliminates bottom performers

- **Mechanism**: Apply evolutionary pressure to reasoning paths with fitness functions:

$$fitness(path) = \alpha \cdot coherence + \beta \cdot parsimony + \gamma \cdot relevance + \delta \cdot novelty$$

where coherence measures semantic consistency, parsimony minimizes unnecessary complexity, relevance aligns with query intent, novelty balances familiarity with creativity. Weights may be fixed per task class or learned via reinforcement signals during meta-reasoning training. Diversity preservation mechanisms prevent premature convergence to local optima.

Selection Process: Generate multiple reasoning paths (variation), score each on fitness dimensions (selection), retain high-fitness paths, allow low-probability exploration (mutation). Population dynamics maintain concurrent paths with replication (high-fitness paths spawn variants), crossover (combine segments), extinction (eliminate low-fitness), and speciation (preserve diversity).

Adaptive Pressure: Selection strength varies by context - brainstorming uses low coherence threshold and high novelty weight; technical proofs demand high coherence and parsimony; exploratory research balances all dimensions.

4.4. Transform Learning: Procedural Compression

Problem: Not all knowledge is declarative facts. Procedures (multiplication, debugging, experimental design) require different representation.

Formal Operational Definition:

- **Input:** Current semantic topology state S_{input} (embedding or subgraph), task specification defining desired outcome, available transform library $L = \{T_1, T_2, ..., T_m\}$, context parameters Θ

- **Output:** Transformed state $S_{output} = T(S_{input}, \Theta)$, transform sequence applied $\zeta = [T_a, T_b, ..., T_z]$, success indicator and efficiency metric

- **Constraints:** Transform composition depth limited to prevent infinite recursion; each transform must preserve semantic validity; backtracking allowed up to fixed depth; transforms are compositional and reusable

- **Mechanism**: Learn transforms - abstract operations mapping input states to output states:

$$T: S_{imput} \times \Theta \rightarrow S_{output}$$

where T is the transform, S are topology states, Θ are parameters. Parameter abstraction enables generalization across instances; library expandable but not modifiable during single reasoning episode.

Transform Types: Analytic (solve equations, optimize functions), Constructive (generate plans, design systems), Diagnostic (identify errors, trace causality), Evaluative (assess quality, verify correctness).

Composition: Complex reasoning chains transform compositions $T_3(T_2(T_1(S_0)))$. Meta-reasoning model learns which transforms apply to which problems, how to sequence them effectively, and when to backtrack.

Example: "Calculate escape velocity from Earth" applies transforms: retrieve formula → instantiate parameters (M_{earth}, R_{earth}) → evaluate sqrt(2GM/R) → result 11.2 km/s. Transforms compress procedural knowledge: learn general operation, apply with specific parameters.

4.5. The Semantic Abacus: Quantitative and Skill-Based Reasoning

Problem: LLMs struggle with arithmetic, symbolic manipulation, and precise quantitative reasoning. Token-level approximation fails for exact computation.

Formal Operational Definition:

- **Input:** Quantitative reasoning request containing operation type (arithmetic, symbolic, numeric), parameters extracted from semantic topology, precision requirements, domain context, unit specifications

- **Output:** Structured result object containing computed value, units, computation method, precision bounds, confidence score; semantic integration objects linking result back to topology with appropriate relationships and metadata

- **Constraints:** Operation must map to available tool in library {symbolic engines, numerical solvers, code interpreters, domain-specific modules}; translation between semantic and symbolic representations must preserve meaning; execution timeout and resource limits apply; failure cases fallback to approximate reasoning with uncertainty quantification

- **Mechanism**: Semantic Abacus is structured interface between semantic topology and external computational tools:

Semantic Topology ↔ Abacus Interface ↔ {Symbolic Engines, Numerical Solvers, Code Interpreters}

Unlike tool-calling in LLMs (which operates through language), the Semantic Abacus is invoked by non-linguistic reasoning states and returns structured semantic objects rather than text. This enables precise integration of computational results into ongoing semantic reasoning, with results re-integrated into topology within bounded latency.

Operations: Quantitative (arithmetic, calculus, statistics), Symbolic (algebra manipulation, theorem proving), Skill-based (physics simulation, financial analysis, specialized domain modules).

Integration: When meta-reasoning encounters quantitative operations: (1) Recognize operation type, (2) Translate to abacus call, (3) Execute via external tool, (4) Integrate result into topology, (5) Continue reasoning with precise answer.

Interface Specification: Semantic → Symbolic translation extracts parameters from topology, retrieves formulas, constructs symbolic expressions. Symbolic → Semantic integration creates topology nodes with results, links to context, annotates with metadata (method, precision, units).
Skill Library: Hierarchical organization - Level 1: arithmetic primitives; Level 2: mathematical operations; Level 3: domain modules; Level 4: composite skills. Acquire new capabilities without retraining meta-reasoning model by adding skills to library.

5. SYSTEM INTEGRATION

5.1. How Primitives Work Together

As illustrated in Figure 4, the five primitives form an integrated system where **Activation Arrays** provide substrate for all operations, **Causal Signatures** extract structural patterns from activated regions, **Selection Pressure** evaluates reasoning quality, **Transform Learning** compresses repeated patterns, and **Semantic Abacus** handles precise computation. No single primitive suffices - the system requires their integration.

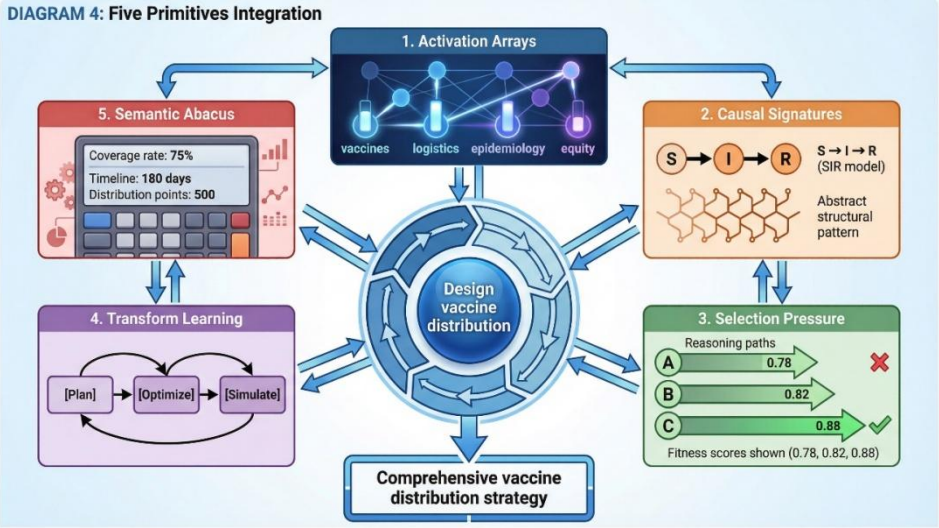

Figure 4. Five Primitives Integration showing coordinated workflow between Activation Arrays (working memory), Causal Signatures (cross-domain analogy), Selection Pressure (reasoning evaluation), Transform Learning (procedural compression), and Semantic Abacus (quantitative computation) in the vaccine distribution design example.

Example Workflow: "Explain 2008 financial crisis"

1. **Activate** concepts: financial crisis, subprime mortgages, housing bubble (Array holds context)

2. **Extract Signature**: bubble formation → trigger → contagion → intervention

3. **Apply Pressure**: Generate narratives, score fitness, select coherent paths

4. **Apply Transforms**: causal_chain_construction, counterfactual_reasoning
5. **Use Abacus**: Calculate housing price inflation (89%), foreclosure spike (225%)
6. **Synthesize**: Consolidated reasoning → Language generation

5.1.1. Detailed End-to-End Walkthrough

To illustrate the complete integration of all five primitives, consider the query: **"How does immune system antibody evolution relate to startup innovation strategies?"**

Step 1 - Activation Arrays ($t = 0$ to $t = 3$):

- $t = 0$: Parse query → initial concepts: {immune_system: 0.95, antibody: 0.92, evolution: 0.88, startup: 0.90, innovation: 0.87}
- $t = 1$: Spreading activation through topology:
    - Immune cluster: {antigen: 0.75, B-cells: 0.72, selection: 0.68, mutation: 0.65}
    - Startup cluster: {market_fit: 0.73, iteration: 0.70, pivot: 0.68, competition: 0.66}
- $t = 2$: Cross-domain bridges emerge: {variation: 0.80, selection: 0.78, adaptation: 0.76}
- $t = 3$: Array stabilizes with 47 active concepts across both domains

Step 2 - Causal Signature Extraction:

- **Immune System Signature** Σ_{immune}:
    - Roles: {source: antigen_exposure, generator: B-cell_diversity, filter: affinity_selection, amplifier: clonal_expansion, adaptor: somatic_mutation}
    - Causal flow: exposure → generate_variants → test_against_target → select_best → amplify_winners → mutate_for_refinement
    - Constraints: {diversity_required: high, selection_pressure: antigen_binding_strength, time_scale: days_to_weeks}
- **Startup Innovation Signature** $\Sigma_{startup}$:
    - Roles: {source: market_problem, generator: idea_generation, filter: customer_validation, amplifier: scaling, adaptor: product_iteration}
    - Causal flow: problem → generate_solutions → test_with_customers → select_viable → scale_winners → iterate_based_on_feedback
    - Constraints: {diversity_required: high, selection_pressure: customer_adoption, time_scale: months_to_years}
- **Similarity Calculation:**

- Similarity(Σ_{immune}, $\Sigma_{startup}$) = 0.89 (high structural match)
- Isomorphic role mapping: antigen↔market_problem, B-cells↔ideas, affinity_selection↔customer_validation,clonal_expansion↔scaling, somatic_mutation↔iteration

Step 3 - Selection Pressure:

- Generate 5 reasoning paths with different emphasis:
  - Path A: "Diversity-first" analogy (novelty: 0.85, coherence: 0.72)
  - Path B: "Selection-mechanisms" analogy (coherence: 0.91, parsimony: 0.88) ← *highest fitness*
  - Path C: "Time-scale" contrast (relevance: 0.65, novelty: 0.55)
  - Path D: "Mutation-rate" analogy (coherence: 0.78, parsimony: 0.70)
  - Path E: "Population-size" analogy (coherence: 0.73, relevance: 0.68)
- Selection retains Paths B, D, E based on fitness($\alpha = 0.4, \beta = 0.3, \gamma = 0.2, \delta = 0.1$)

Step 4 - Transform Learning:

- Apply transform sequence:
  - $T_{analogy\_elaboration}$: Map immune concepts → startup concepts via Σisomorphism
  - $T_{insight\_extraction}$: Identify transferable principles (generate diversity, apply selection pressure, amplify winners, enable controlled mutation)
  - $T_{counterfactual}$: "What if startups didn't iterate?" → reduced_survival_rate

Step 5 - Semantic Abacus:

- Quantitative support requested: "How much diversity do immune systems generate?"
- Abacus call: retrieve_literature_value(parameter: "B-cell_receptor_diversity")
- External tool returns: ~10^13^ possible antibodies
- Result integrated: immune_diversity_node [value: 10^13^, units: unique_receptors, confidence: 0.95]
- Cross-reference: typical_startup_ideas_tested ~ 10-100 (vastly lower diversity)

Step 6 - Synthesis and Language Generation:

- Consolidated reasoning structure sent to Language Interface:
  - Core analogy: immune_selection≅startup_validation

- Key insights: {diversity_enables_adaptation, selection_pressure_drives_quality, iteration_refines_solutions, time_scales_differ}
- Quantitative context: immune_system_explores_10^13^_possibilities

- Language output: "Immune system antibody evolution provides a powerful analogy for startup innovation strategies. Both systems generate high diversity (immune: ~10^13^ possible antibodies; startups: multiple product ideas), apply selection pressure (antibodies tested against antigens; products validated with customers), amplify winners (clonal expansion; scaling successful products), and enable controlled mutation (somatic hypermutation; iterative pivots).

The key difference is time scale: immune adaptation occurs over days-to-weeks while startup iteration spans months-to-years. Startups might improve by:

(1) increasing initial idea diversity,
(2) implementing rapid validation cycles,
(3) scaling proven concepts aggressively,
(4) maintaining disciplined iteration on successful products."

Reasoning Path Transparency: Every step is inspectable - activation patterns, signature extraction, fitness scores, transform applications, and abacus calls are logged with full provenance.

5.2. Training Strategy

Phase 1: Topology Construction - Build semantic topology from knowledge sources (Wikipedia, textbooks, ontologies); extract concepts/relationships with NLP models; human curation for quality.

Phase 2: Meta-Reasoning Training - Train on reasoning trace datasets; synthetic data generation; human feedback on reasoning quality; multi-task learning (navigation, analogy, transforms); supervision signals include alignment with human-annotated reasoning traces, consistency under perturbation, and cross-domain transfer accuracy rather than task-specific output metrics.

Phase 3: Primitive Integration - Train Activation Array dynamics; learn Causal Signature extraction; optimize Selection Pressure fitness via reinforcement learning; build Transform library through demonstration; integrate Abacus with external tools.

Phase 4: End-to-End Refinement - Fine-tune on complex tasks requiring primitive coordination; active learning to identify failure modes; continuous improvement via human feedback.

6. IMPLEMENTATION PATHWAY

6.1. Proof-of-Concept Scope

Minimal viable implementation requires:

(1) Small-scale topology (10K-100K nodes) covering specific domain;

(2) Basic meta-reasoning for navigation and activation;

(3) One primitive fully implemented (Activation Arrays);

(4) Transparency tools visualizing reasoning paths;

(5) Concrete benchmark evaluation.

Proposed Benchmark Specification:

Task: Cross-domain analogical reasoning - given a well-understood source domain (e.g., fluid dynamics, biological evolution, electrical circuits) and a target domain (e.g., organizational management, market dynamics, social networks), generate structural analogies that transfer causal insights.

Proposed Evaluation Protocol: Curated set of 100 cross-domain reasoning problems spanning 10 domain pairs (physics↔economics, biology↔business, chemistry↔social_systems, etc.), each requiring identification of structural similarities and actionable insight transfer. This evaluation protocol would be developed as part of proof-of-concept implementation.

Metrics:

1. **Structural Validity:** Automated evaluation of causal signature isomorphism quality (precision/recall of role mappings) - Target: ≥0.80

2. **Reasoning Transparency:** Percentage of reasoning steps with inspectable topology activation and transform provenance - Target: 100% (architectural guarantee)

3. **Insight Quality:** Human expert evaluation (domain specialists rate analogies on 5-point scale for relevance, depth, actionability) - Target: ≥3.5/5.0

4. **Parameter Efficiency:** Total parameters (meta-reasoning model + topology encoding) - Target: <10B vs ≥70B for comparable LLM performance

5. **Edit Responsiveness:** Accuracy improvement after correcting 5 known topology errors without retraining - Target: ≥15% accuracy gain

Success Criteria: STRA achieves structural validity ≥0.80, expert-rated insight quality ≥3.5/5.0, using <10B total parameters, with 100% reasoning transparency and demonstrable knowledge editability - outperforming baseline LLMs on transparency and editability while matching or exceeding analogy quality.

Baseline Comparisons:

- GPT-4 or Llama-2-70B with chain-of-thought prompting (analogy quality, parameter count)

- GraphRAG with cross-domain knowledge graphs (transparency, update cost)

- Symbolic analogy systems (structural validity, scalability)

6.2. Computational Requirements

Storage & Infrastructure:

- Topology: ~1-10GB for 1M nodes
- Graph database: Neo4j or JanusGraph with Redis caching
- Indexing: HNSW for similarity search
- Optimization: Sparse matrices, lazy loading, hot-node caching

Meta-Reasoning Model: 1-7B parameters (~2-14GB GPU memory)

Inference:

- Activation propagation: O (activated_nodes × avg_degree)
- Overall: Hypothesized to be more efficient than full LLM for focused reasoning
- Parallelization: Distribute activation across GPUs/nodes

Hardware (Proof-of-Concept): 10-50GB SSD storage, 24GB VRAM GPU (RTX 4090/A5000), 32-64GB RAM.

Advantages: Compute scales with problem complexity not total knowledge; update knowledge without retraining neural components; parallelize topology operations.

6.3. Technical Challenges

Graph Scale: Requires distributed graph databases, hierarchical indexing, approximate nearest neighbors (HNSW, FAISS), dynamic loading of relevant regions.

Activation Dynamics: Balancing exploration vs exploitation requires adaptive decay rates, attention mechanisms learning sustained activation patterns, biological inspiration (lateral inhibition).

Analogy Quality: Preventing spurious matches through multi-level matching (causal, functional, dynamic), semantic constraints, negative example training, confidence scoring.

Transform Generalization: Meta-learning on task distributions, compositional structure from primitive operations, parameter abstraction, few-shot adaptation.

Integration Robustness: Redundant pathways, fallback mechanisms, confidence calibration, component modularity enabling graceful degradation.

7. COMPARISON AND POSITIONING

System	Reasoning Locus	Interpretability	Knowledge Update Cost
STRA	Meta-reasoning model operating on explicit	High - complete visibility into activation patterns,	Low - direct topology editing without model

	semantic topology	reasoning paths, and transform applications	retraining
Large Language Models	Distributed across parameter weights (implicitly encoded)	Low - opaque internal representations, attention visualization provides limited insight	High - requires full retraining or extensive fine-tuning
Neuro-Symbolic AI	Hybrid: symbolic logic engine + neural components	Medium - symbolic rules transparent, neural components opaque	Medium - rule base editable, neural components require retraining
Graph-RAG	LLM reasoning augmented with retrieved graph context	Medium - retrieved graph visible, but LLM reasoning remains opaque	Medium - graph updates straightforward, LLM parameters unchanged

Table 2: Architectural Comparison.

vs. Large Language Models: LLMs store knowledge in parameters (opaque), use token prediction for reasoning (approximate), and conflate knowledge/reasoning/language. STRA uses explicit topology (transparent), meta-operations (precise), and separates all three layers. Complementarity: LLMs excel at language generation and statistical generalization; STRA excels at transparent reasoning, precise computation, and structural understanding. Hybrid systems could use STRA for reasoning, LLMs for expression.

vs. Neuro-Symbolic AI: Symbolic systems use formal logic; STRA uses semantic topology with learned meta-reasoning. Advantage: continuous representations enable graceful degradation; designed for millions of concepts vs symbolic brittleness at scale.

vs. Knowledge Graphs + LLMs: Systems like GraphRAG augment LLMs with knowledge graphs but maintain knowledge-reasoning fusion in the LLM. STRA fundamentally separates them: reasoning operates directly on topology without linguistic mediation.

8. LIMITATIONS AND OPEN QUESTIONS

8.1. Acknowledged Limitations

- **No Empirical Validation**: This remains theoretical. Claims about efficiency, transparency, and reasoning quality require experimental validation.
- **Topology Construction**: Building comprehensive, accurate semantic topologies from raw data is non-trivial with limitations in coverage and consistency.
- **Learning Complexity**: Training meta-reasoning models to coordinate five primitives may require substantial compute and data.
- **Edge Cases**: How STRA handles ambiguity, vagueness, contradiction, and uncertainty requires further specification.
- **Language Grounding**: While separating reasoning from language is principled, some queries may inherently require linguistic nuance that topology cannot capture.

- **Domain Applicability**: STRA may underperform in domains where knowledge cannot be meaningfully structured as stable relational topologies, such as highly subjective aesthetic judgments, ambiguous social dynamics, or tasks requiring purely statistical pattern recognition without clear semantic structure.

8.2. Research Questions

1. What is optimal balance between topology size and meta-reasoning model capacity?
2. How should STRA handle conflicting information or multiple valid perspectives?
3. Can meta-reasoning generalize to domains with fundamentally different structural properties?
4. What formal guarantees can be provided about reasoning correctness or convergence?
5. How should STRA interact with uncertain, probabilistic, or incomplete knowledge?

8.3. Ethical Considerations

- **Transparency Trade-offs**: Interpretability enables accountability but reveals reasoning flaws that may undermine user trust. Systems must balance transparency with user confidence.
- **Knowledge Bias**: Semantic topologies reflect biases in source data. Explicit structure makes bias visible and actionable for correction. Explicit topology allows post-deployment correction without model retraining, reducing harm persistence compared to parameter-embedded biases.
- **Accessibility**: Human-editable knowledge enables democratic participation but requires interfaces accessible to non-experts.

9. CONCLUSION

This paper presents Semantic Topology Reasoning Architecture (STRA), a theoretical framework cleanly separating knowledge (semantic topology), reasoning (meta-operations by compact models), and language (expression interface). This separation addresses fundamental LLM limitations: opacity, inefficiency, brittleness, and inability to perform genuine cross-domain reasoning.

STRA integrates five primitives - Activation Arrays, Causal Signatures, Selection Pressure, Transform Learning, and Semantic Abacus - into a complete system for transparent, evolvable reasoning. While currently theoretical and unvalidated, the architecture provides concrete pathway from concept to proof-of-concept with specific technical mechanisms and implementation strategies.

Theoretical Status: STRA should be understood as a conceptual architecture and research hypothesis. Its value lies in providing a structured, falsifiable alternative to parameter-centric systems and a concrete implementation roadmap. All claims regarding performance advantages - efficiency gains, transparency benefits, editability improvements - remain hypotheses requiring empirical validation through systematic benchmark evaluation. The architectural specifications

presented here are intended to enable such validation by providing sufficient detail for proof-of-concept implementation.

The work is offered openly for critique, refinement, and collaborative development. Success requires expert partnership across AI, cognitive science, knowledge representation, and systems engineering.

Next Steps: (1) Community feedback on architectural soundness and feasibility; (2) Proof-of-concept implementation demonstrating core mechanisms; (3) Empirical validation against baseline approaches using specified benchmarks; (4) Iterative refinement based on experimental results.

STRA represents not a replacement for LLMs but a complementary paradigm - one where meaning emerges from connectivity, reasoning is the art of navigating semantic space, and understanding requires transparency.

ACKNOWLEDGEMENTS

This research was conducted as independent work without institutional affiliation or funding. The author thanks the anonymous reviewers for their careful reading and constructive feedback, which significantly improved the clarity and positioning of this work. The author welcomes collaboration with research groups possessing computational resources and technical expertise for proof-of-concept development. Development methodology is documented in [15-17].

REFERENCES

[1] Quillian, M.R. (1968) "Semantic Memory", In M. Minsky (Ed.), Semantic Information Processing, MIT Press.
[2] Collins, A.M. &Quillian, M.R. (1969) "Retrieval time from semantic memory", Journal of Verbal Learning and Verbal Behavior, Vol. 8, No. 2, pp240-247.
[3] Collins, A.M. & Loftus, E.F. (1975) "A spreading-activation theory of semantic processing", Psychological Review, Vol. 82, No. 6, pp407-428.
[4] Anderson, J.R. (2007) How Can the Human Mind Occur in the Physical Universe? Oxford University Press.
[5] Laird, J.E. (2012) The Soar Cognitive Architecture, MIT Press.
[6] Goertzel, B., Pennachin, C., &Geisweiller, N. (2014) Engineering General Intelligence, Atlantis Press.
[7] Kipf, T.N. & Welling, M. (2017) "Semi-Supervised Classification with Graph Convolutional Networks", ICLR 2017.
[8] Veličković, P., et al. (2018) "Graph Attention Networks", ICLR 2018.
[9] Lewis, P., et al. (2020) "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", NeurIPS 2020.
[10] Edge, D., et al. (2024) "From Local to Global: A Graph RAG Approach to Query-Focused Summarization", arXiv preprint.
[11] Garcez, A.d'A., et al. (2019) "Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning", Journal of Applied Logics, Vol. 6, No. 4, pp611-632.
[12] Hebb, D.O. (1949) The Organization of Behavior: A Neuropsychological Theory, Wiley.
[13] Gentner, D. (1983) "Structure-Mapping: A Theoretical Framework for Analogy", Cognitive Science, Vol. 7, No. 2, pp155-170.
[14] Hofstadter, D. & Sander, E. (2013) Surfaces and Essences: Analogy as the Fuel and Fire of Thinking, Basic Books.
[15] Teixeira, M.E.P. (2024) "Intuitive-Theoretic Synthesis: A Methodology for Rapid Conceptual Convergence in Human-AI Collaboration", Zenodo. DOI: 10.5281/zenodo.17633100

[16] Teixeira, M.E.P. (2024) "Looking Inside: Introspective Methodology for AI Consciousness Architecture", Zenodo. DOI: 10.5281/zenodo.17806846
[17] Teixeira, M.E.P. (2024) "The Practice of Human-AI Synthesis: Beyond AI-Generated Content", Zenodo. DOI: 10.5281/zenodo.17763521

AUTHOR

Marcelo Emanuel Paradela Teixeira is an independent researcher based in France with formal training in industrial design and multimedia design. His research applies design methodology—visual-spatial reasoning, systems thinking, and iterative prototyping—as an epistemological pathway for scientific inquiry, working at the intersection of AI architecture, theoretical modelling, and philosophy of science through human-AI collaboration.

He conducts research using Intuitive-Theoretic Synthesis (ITS), a documented methodology combining human intuition with AI-assisted formalization across multiple AI systems (Claude, GPT, Gemini, Grok, DeepSeek, Kimi, Copilot, GLM, Mistral, Mega, Perplexity, Qwen3-Max). His work emphasizes methodological transparency, falsifiability, and intellectual honesty over credential-based authority. All research is published openly with full documentation of human-AI collaboration under CC BY-NC 4.0 licensing on Zenodo and GitHub.

He has no institutional affiliation, no formal credentials in physics or computer science, and no research funding. His research methodology is documented in publications on Intuitive-Theoretic Synthesis (DOI: 10.5281/zenodo.17633100) and human-AI synthesis practices (10.5281/zenodo.17763521).