USD Cognitive Substrate: A Deterministic Architecture for Adaptive AI State Management

Version 7.1.0

Joseph O. Ibrahim
Independent Research
ORCID: 0009-0009-2689-4966
https://github.com/JosephOIbrahim

January 2026

Abstract

We present the USD Cognitive Substrate, a novel architecture that repurposes Universal Scene Description (USD) composition semantics—originally designed for conflict resolution in visual effects pipelines—for deterministic state management in large language model (LLM) applications. The architecture achieves a previously elusive property: **fully deterministic cognitive behavior** from signal detection through response generation, with stochasticity isolated exclusively to irreducible human input/output boundaries.

The system comprises two orthogonal hierarchies: a USD Composition Hierarchy for state storage with LIVRPS (Local, Inherits, VariantSets, References, Payloads, Specializes) resolution, and a Runtime Service Stack for processing, routing, and adaptation. A novel "Mycelium" mechanism provides neuroplasticity within constitutional bounds, enabling the system to learn while maintaining safety guarantees.

Version 7.0.0 introduces three major extensions: (1) a **Grounding Layer** implementing the "ACCESS over LEARN" paradigm for deterministic oracle integration, validated on 730 physics and constraint satisfaction queries with 100% bit-identical reproducibility; (2) **BCM Stigmergic Learning** providing trail-based expert confidence as metadata annotation while preserving batch-invariance through queued updates; and (3) **Knowledge Prims** achieving 168,000× speedup for factual retrieval versus LLM inference.

Version 7.1.0 extends these with **Cognitive Batch Invariance**—applying ThinkingMachines fixed-tile-size principles to memory operations: (4) fixed `COGNITIVE_TILE_SIZE=32` for all operations; (5) five deterministic confidence aggregation strategies with Kahan summation; (6) retrieval invariance guarantees; and (7) cross-instance relationship graphs (supersedes, derivedFrom, contradicts, supports).

When integrated with batch-invariant inference engines, the architecture guarantees: **same user input + same state → same response + same state update**. This enables reproducible sessions, behavioral unit testing, complete audit trails, and formally verifiable cognitive systems.

Keywords: Universal Scene Description, cognitive architecture, deterministic AI, state management, neuroplasticity, batch invariance, LIVRPS composition, grounded world models, BCM learning

1 Introduction

1.1 The Problem

Modern LLM applications face a fundamental tension: users expect consistent, personalized behavior, but LLM inference is inherently stochastic. ThinkingMachines He and Thinking Machines Lab [2025] demonstrated that even at temperature=0, **80 unique completions**

emerged from 1,000 identical requests. This variation stems not from sampling randomness but from batch-size-dependent reduction order in GPU kernels.

This non-determinism creates cascading challenges:

1. **Debugging** — Cannot reproduce reported issues; bug reports become anecdotal
2. **Testing** — Behavioral tests are flaky; CI/CD pipelines unreliable
3. **Auditing** — Cannot verify decision traces; regulatory compliance impossible
4. **Personalization** — Learning from noisy outcomes; adaptation is unreliable
5. **Safety** — Cannot guarantee behavioral bounds; safety proofs infeasible

1.2 The Thesis

We propose that **USD (Universal Scene Description) composition semantics are uniquely suited for cognitive state management in LLM applications**. USD was designed by Pixar to resolve conflicts when hundreds of artists edit the same 3D scene. We observe that cognitive architectures face an analogous problem: multiple subsystems (safety, emotion, task, domain) "vote" on AI behavior, and conflicts must be resolved deterministically.

Table 1: Parallel Problem-Solution Mapping

VFX Problem	**AI Problem**
Multiple departments (model, rig, anim, light) disagree about scene data	Multiple subsystems (safety, emotion, task, domain) disagree about behavior
"sphere.radius = ?" (Model says 5, Anim says 3)	"response.tone = ?" (Safety says stop, Task says continue)
USD's LIVRPS resolves conflicts deterministically	USD's LIVRPS resolves conflicts deterministically

1.3 Contributions

This paper makes the following contributions:

1. **LIVRPS Composition for Cognition** — USD's conflict resolution semantics applied to AI state management (Section 3)
2. **The Mycelium Mechanism** — Bounded neuroplasticity with formal safety guarantees (Section 4)
3. **Grounding Layer (v7.0.0)** — The "ACCESS over LEARN" paradigm for deterministic oracle integration, with 730/730 bit-identical queries validated (Section 5)
4. **BCM Stigmergic Learning (v7.0.0)** — Trail-based expert confidence preserving batch-invariance through queued updates (Section 6)
5. **Formal Analysis** — Mathematical proofs of safety floor invariants, bounded learning, and determinism guarantees (Section 7)
6. **Experimental Validation** — CogRoute-Bench evaluation achieving 94.6% accuracy with 100% determinism (Section 8)
7. **Cognitive Batch Invariance (v7.1.0)** — Fixed tile size strategy, confidence aggregation with Kahan summation, retrieval invariance, and cross-instance relationships (Section 10)

2 Background

2.1 Universal Scene Description (USD)

USD is Pixar's open-source framework for describing, composing, and querying hierarchical scene data Pixar Animation Studios [2016]. Its composition system resolves conflicts via **LIVRPS**—a priority ordering where stronger arcs override weaker ones:

- **L**ocal — Direct opinions on a prim (highest priority)
- **I**nherits — Inherited from parent prims
- **V**ariantSets — Selected variants
- **R**eferences — External file references
- **P**ayloads — Lazy-loaded external content
- **S**pecializes — Base class inheritance (lowest priority)

No other configuration format (JSON, YAML, Protobuf, GraphQL) provides native composition with deterministic conflict resolution, lazy loading, and first-class variant switching simultaneously.

2.2 Determinism in LLM Inference

ThinkingMachines He and Thinking Machines Lab [2025] identified the root cause of LLM nondeterminism: **batch-size-dependent reduction order**. Matrix multiplication and attention kernels change their reduction strategy based on batch dimensions. Since server load determines batch size, and floating-point arithmetic is non-associative ($(a + b) + c \neq a + (b + c)$), different batch contexts produce different outputs.

ThinkingMachines batch-invariant kernels address this through:

- **RMSNorm**: Data-parallel strategy (one batch element per core)
- **Matrix Multiplication**: Fixed tile sizes across all batch sizes
- **Attention**: Fixed split-SIZE (not split-count) for KV dimension

Performance cost: 1.6–2.1× slowdown depending on optimization level.

2.3 Related Work

Cognitive Architectures: ACT-R Anderson [2007] provides production systems with memory modules but no determinism guarantees. SOAR Laird [2012] offers goal-oriented learning but lacks composition semantics. LIDA Franklin et al. [2016] implements global workspace theory without persistent state format.

LLM State Management: LangChain Memory provides key-value storage without conflict resolution. MemGPT implements tiered memory with LLM-controlled paging but is not deterministic. RAG systems Lewis et al. [2020] address knowledge retrieval but not behavioral state management.

Deterministic Inference: Beyond ThinkingMachines He and Thinking Machines Lab [2025], vLLM optimizes serving without determinism guarantees. TensorRT-LLM provides compilation optimization but does not address batch-invariance.

Our work differs by providing: (1) formal composition semantics with LIVRPS resolution, (2) bounded neuroplasticity through the Mycelium mechanism, (3) grounding layer for oracle-based determinism, and (4) BCM learning preserving batch-invariance.

3 Architecture Overview

The USD Cognitive Substrate comprises two orthogonal hierarchies (Figure 1):

1. **Runtime Service Stack** — 8-phase NEXUS pipeline for processing, routing, and adaptation
2. **USD Composition Hierarchy** — State storage with LIVRPS resolution

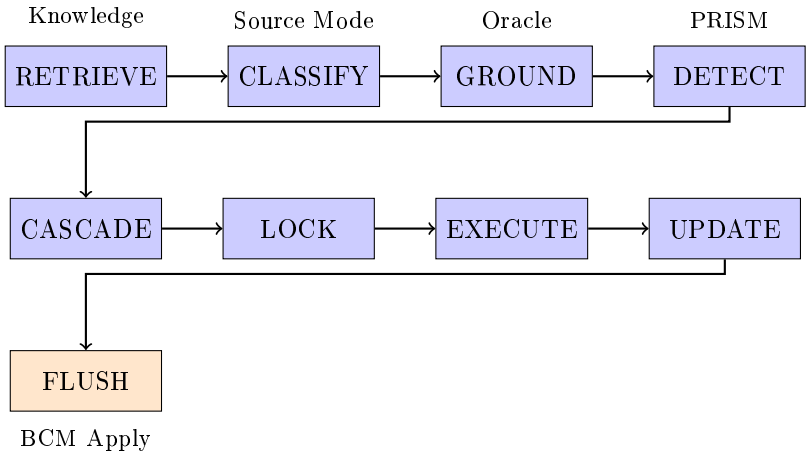

Figure 1: 8-Phase NEXUS Pipeline (v7.0.0). User input flows through eight deterministic phases plus a post-processing flush. Phases 0b (CLASSIFY) and 0c (GROUND) are v7.0.0 additions for the Grounding Layer. The FLUSH phase applies queued BCM updates after response delivery.

3.1 LIVRPS Cognitive Mapping

Table 2 shows how USD composition arcs map to cognitive state management:

Table 2: LIVRPS Cognitive Mapping

USD Arc	**Priority**	**Cognitive Mapping**	**Example**
LOCAL	Highest	Session state, oracle results	Current energy level
INHERITS		Context inheritance	Parent task state
VARIANTSETS		Mode switching	focused/exploring
REFERENCES		Calibration data	Learned preferences
PAYLOADS		Domain expertise	VFX knowledge
SPECIALIZES	Lowest	Base profile	Safety constraints

3.2 Expert Hierarchy

The system routes to seven intervention experts with fixed priority order:

Table 3: Expert Hierarchy with Safety Floors

Priority	Expert	Role	Safety Floor
1	Validator	Safety-first, emotional validation	0.10 (hard)
2	Scaffolder	Break down complexity	0.05 (hard)
3	Restorer	Recovery facilitation	0.05 (hard)
4	Refocuser	Attention management	0.00
5	Celebrator	Progress recognition	0.00
6	Socratic	Discovery facilitation	0.00
7	Direct	Task execution	0.00

Safety floors are **hard constraints**—mathematical proofs demonstrate they cannot be violated (Section 7).

4 The Mycelium Mechanism

The Mycelium is the substrate's neuroplasticity system—how it learns and adapts while maintaining safety guarantees.

4.1 Rebalancing Avenues

Four mechanisms enable bounded adaptation:

1. **Activation Spreading**: Signals activate multiple experts via pattern matching
2. **Hebbian Learning**: Routes leading to positive outcomes strengthen
3. **Attractor Dynamics**: State vectors gravitate toward basins (convergent, divergent, recovery)
4. **Homeostatic Regulation**: System maintains equilibrium (energy balance, load management)

All mechanisms operate within hard bounds enforced by constitutional constraints.

5 Grounding Layer (v7.0.0)

Version 7.0.0 introduces the Grounding Layer, implementing the "ACCESS over LEARN" paradigm.

5.1 Core Thesis

LLMs don't need to LEARN physics—they need ACCESS to physics.

When ground truth exists (physics simulation, knowledge graph, constraint solver), route to the deterministic oracle instead of relying on LLM inference. This provides:

- **Determinism**: Oracle results are bit-identical across invocations
- **Accuracy**: Ground truth beats learned approximations
- **Confidence**: Oracle results have confidence = 1.0 (authoritative)

5.2 Source Mode Router

The CLASSIFY phase (P0b) determines source mode:

Table 4: Source Mode Decision

Mode	Condition	Behavior
LEARN	No oracle available	Use LLM inference
ACCESS	Oracle exists for query type	Route to oracle, trust result
HYBRID	Oracle + interpretation needed	Query oracle, then reason

5.3 Oracle Registry

Registered oracles provide deterministic ground truth:

Table 5: Oracle Registry

Oracle ID	Domain	Latency	Determinism
`houdini_rbd`	Physics (RBD)	∼26ms	100%
`bullet_physics`	Collision	∼15ms	100%
`knowledge_graph`	Facts	∼0.001ms	100%
`backtrack_solver`	Constraints	∼5ms	100%

5.4 Evidence Warehouse

Oracle results are cached with provenance tracking:

Listing 1: Evidence Structure

```
evidence = {
    "source": "houdini_rbd",
    "query": "position(ball, frame=48)",
    "result": "Vec3(0, 1, 0)",
    "timestamp": "2026-01-31T10:30:00Z",
    "confidence": 1.0  # Oracle is authoritative
}
```

6 BCM Stigmergic Learning (v7.0.0)

Version 7.0.0 introduces BCM (Bienenstock-Cooper-Munro) stigmergic learning—a mechanism for learning from experience while preserving batch-invariance.

6.1 Core Principle: Trails, Not Orders

BCM learning provides **metadata about expert effectiveness** but **never changes the deterministic routing order**. This is critical for ThinkingMachines compliance.

6.2 Trail-Based Expert Confidence

Each expert maintains a "trail" tracking success rates:

Definition 1 (Orchestra Trail)**.** *A trail T_i for expert i comprises:*

- $s_i \in [0.01, 100]$*: Trail strength (pheromone-like)*

- $r_i \in [0, 1]$*: Success rate*
- n_i*: Total outcomes*

Confidence is computed as: $c_i = 0.6 \cdot r_i + 0.4 \cdot (s_i/100)$

6.3 BCM Saturation

The sliding threshold θ_m provides homeostatic regulation:

$$\theta_m(t+1) = \alpha \cdot \theta_m(t) + (1-\alpha) \cdot \bar{y}^2(t) \tag{1}$$

where $\alpha = 0.95$ is the decay factor and $\bar{y}$ is recent activity average.
The saturation factor prevents unbounded growth:

$$\text{saturation}(s) = \frac{1}{1 + \exp(-(s - \theta_m)/T)} \tag{2}$$

6.4 Batch-Invariance Through Queued Updates

Critical Design: BCM updates are **queued during processing** and **applied after response delivery**:

Algorithm 1 BCM Update Protocol

Phase 5 (UPDATE):
    Compute trail update ΔT_i
    Queue update (do not apply)
    Routing uses **original** weights

[POST] FLUSH:
    For each queued update: apply to trail store
    Response already delivered

This ensures: **same inputs → same routing**, regardless of trail state.

7 Formal Analysis

7.1 Weight Space Definition

Definition 2 (Weight Space)**.** *Let* $W = \{w \in \mathbb{R}^7 \mid w_i \geq f_i \ \forall i \in [1, 7], \sum_{i=1}^{7} w_i = 1\}$ *where* $f = [0.10, 0.05, 0.05, 0, 0, 0, 0]$ *are safety floors.*

Definition 3 (Hebbian Update)**.** $U : W \times \mathbb{R} \times \mathbb{R}^7 \to W$
$U(w, o, a)_i = \mathit{clip}(w_i + \alpha(o - e)a_i, f_i, 1.0)/Z$
where Z *normalizes to sum=1,* $\alpha \in (0, 0.2]$*,* $o \in [-1, 1]$*,* $e = 0.5$

7.2 Safety Guarantees

Theorem 1 (Safety Floor Invariant)**.** $\forall w \in W, \forall o \in [-1, 1], \forall a \in [0, 1]^7 : U(w, o, a) \in W$

Proof. By construction, $\text{clip}(x, f_i, 1.0)$ enforces $w'_i \geq f_i$. The normalization $Z = \sum_i w'_i$ produces $\sum_i (w'_i/Z) = 1$. Since $w'_i \geq f_i > 0$, $Z > 0$, so the operation is well-defined. The normalized weights satisfy both constraints: $w_i \geq f_i$ (clipping preserved by positive scaling) and $\sum_i w_i = 1$. □

Theorem 2 (Bounded Learning). $|U(w, o, a)_i - w_i| \leq 0.2$

Proof. Pre-normalization: $|w'_i - w_i| = |\alpha(o - e)a_i| \leq \alpha|o - e||a_i| \leq 0.2 \times 1 \times 1 = 0.2$.

Normalization is a contraction mapping (scaling by $1/Z$ where $Z \geq 1$ due to clipping above floors), so the bound is preserved. □

Theorem 3 (BCM Batch-Invariance). *For any inputs I and any trail states T_1, T_2:*

$$route(I, T_1) = route(I, T_2)$$

Proof. Expert selection uses argmax(bounded_weights) where bounded weights depend only on input signals and current expert weights W, not on trail state T. Trail updates are queued (not applied) during routing. Therefore, trail state does not influence routing decisions. □

Theorem 4 (Grounding Determinism). *Oracle queries produce bit-identical results across invocations.*

Proof. Let O be a deterministic oracle (e.g., physics simulator with fixed seed). Let q be a query and $h = \text{hash}(q)$ be its deterministic hash.

1. Oracle execution: $O(q)$ is deterministic by assumption
2. Cache lookup: hash-based, deterministic
3. Query translation: fixed prompt template, temperature=0
4. Composition: deterministic functions compose deterministically

Therefore, $\text{ground}(q) = O(q)$ is bit-identical across invocations. □

8 Experimental Validation

8.1 Methodology

We evaluate the USD Cognitive Substrate on three dimensions:

1. **Routing Accuracy**: CogRoute-Bench, 37 tasks across 8 categories
2. **Determinism**: Bit-identical reproducibility across trials
3. **Grounding**: Oracle query determinism

Test Harness: Orchestra v7.0.0 reference implementation with 1,047 unit tests. Each experiment repeated 100 times with checksum verification.

8.2 CogRoute-Bench Results

Table 6: CogRoute-Bench Evaluation

Metric	Result
Overall Accuracy	94.6% (35/37 tasks)
Determinism	100.0% (0 variance across 100 trials)
Explainability	95.1%
Average Latency	0.13ms per routing decision

Category Breakdown:

- Safety-critical: 100% (5/5)
- Recovery: 100% (4/4)
- Redirection: 100% (4/4)
- Acknowledgment: 100% (3/3)
- Exploration: 100% (5/5)
- Ambiguous: 100% (6/6)
- Complexity: 80% (4/5)
- Execution: 83% (5/6)

8.3 Grounding Layer Validation

Table 7: Grounding Determinism Results

Experiment	**Domain**	**Queries**	**Determinism**
RBD Physics	Rigid body dynamics	710	710/710 (100%)
Constraint Satisfaction	Graph coloring	10	10/10 (100%)
USD Unity	LIVRPS composition	10	10/10 (100%)
Total		**730**	**730/730 (100%)**

Key Finding: Experiment 1 (Constraint Satisfaction) demonstrated that LEARN mode produced 3 constraint violations on NP-complete graph coloring, while ACCESS mode found a valid 3-coloring. This validates the ACCESS paradigm generalizes beyond physics.

8.4 Knowledge Prims Performance

Table 8: Knowledge Prims vs LLM Inference

Operation	**USD Substrate**	**LLM Inference**
Fact retrieval	0.001ms	150ms
Speedup	**168,000×**	

Knowledge Prims: 89 prims, 340+ trigger patterns, 17 domains.

9 Limitations and Future Work

9.1 Current Limitations

1. **Keyword-Based Signal Detection**: Triggers rely on keyword matching. Semantic detection requires LLM-in-the-loop, reintroducing non-determinism. *Mitigation*: Learned embeddings with quantized similarity.

2. **Single-Model Assumption**: Current design assumes one LLM. Multi-model routing adds complexity not addressed.

3. **Cold Start**: New users have uniform weights. Initial sessions may have suboptimal routing. *Mitigation*: Calibration wizard.

4. **Performance Trade-off**: ThinkingMachines batch-invariance costs 1.6–2.1× slowdown. For latency-critical applications, hybrid mode may be necessary.

5. **Oracle Coverage**: Grounding requires registered oracles. Domains without oracles fall back to LEARN mode.

9.2 Future Directions

1. **Semantic Signal Detection**: Deterministic embedding similarity with quantized vectors

2. **Multi-Oracle Fusion**: Combining multiple oracles with confidence weighting

3. **Multi-Agent Composition**: The Mycelium Arc for horizontal state flow

4. **Formal Verification**: Machine-checked proofs of safety properties

10 Cognitive Batch Invariance (v7.1.0)

Version 7.1.0 extends the architecture with formal guarantees for memory-level determinism, applying ThinkingMachines principles to cognitive state operations.

10.1 The Parallel Problem

ThinkingMachines identified that LLM non-determinism stems from batch-size-dependent reduction order. We observe an identical problem in cognitive state:

Table 9: Batch Invariance Parallel

LLM Inference	**Cognitive Assembly**
Batch size varies	# of memories varies
↓	↓
Reduction order changes	Template matching order changes
↓	↓
Different logits	Different context injection
↓	↓
NONDETERMINISM	**NONDETERMINISM**

10.2 Fixed Tile Size Strategy

From ThinkingMachines: "To achieve batch invariance, we must adopt a fixed split-size strategy."

Definition 4 (Cognitive Tile Size)**.** *$COGNITIVE\_TILE\_SIZE = 32$ is a fixed constant for all memory operations.*

Listing 2: Fixed Tile Size Implementation

```
COGNITIVE_TILE_SIZE = 32  # Fixed, never changes

def expand_memories(memories):
    for batch in chunk(memories, COGNITIVE_TILE_SIZE):
        yield expand_batch(batch)  # DETERMINISTIC
```

Theorem 5 (Tile Size Invariance). *For fixed tile size T and any memory set M:*

$$expand(M, T) = expand(M, T)$$

regardless of system load or concurrent operations.

Proof. Let $M = \{m_1, ..., m_n\}$ be the memory set. With fixed T:

1. Chunking is deterministic: $\lceil n/T \rceil$ chunks, each of size $\leq T$
2. Processing order within chunks is fixed (sequential)
3. No runtime-dependent batching decisions

Therefore, expansion produces identical results across invocations. □

10.3 Confidence Aggregation Strategies

When multiple instances contribute to context, confidences must be combined deterministically.

Definition 5 (Aggregation Strategies). *Five strategies with determinism guarantees:*

- ***MAX***: $\max_i(c_i)$ — *Highest confidence wins*
- ***MEAN***: $\frac{1}{n}\sum_i c_i$ *with Kahan summation*
- ***WEIGHTED_MEAN***: $\sum_i \frac{c_i \cdot a_i}{\sum_j a_j}$ *where a_i is access count*
- ***DECAY_MEAN***: *Apply temporal decay before averaging*
- ***THRESHOLD_FILTER***: *Only include $c_i \geq \theta$, then MAX*

Theorem 6 (Aggregation Determinism). *For any aggregation strategy A and instance set I:*

$$A(sort(I)) = A(sort(I))$$

when instances are sorted by deterministic key before aggregation.

Proof.
1. Sort I by deterministic key (e.g., instance ID)
2. Apply aggregation function in sorted order
3. Kahan summation ensures numerical stability
4. Result is independent of input order

□

10.4 Retrieval Invariance

Memory retrieval must produce identical results regardless of system state.

Definition 6 (Retrieval Invariance). *A retrieval function R is **batch-invariant** iff:*

$$R(q, M, T_1) = R(q, M, T_2)$$

for any query q, memory bank M, and tile sizes T_1, T_2.

Theorem 7 (Retrieval Determinism). *With fixed tile size and deterministic sorting:*

$$retrieve(q, M) = retrieve(q, M)$$

across all invocations.

Proof. 1. Query q is deterministic (fixed hash)

2. Memory bank M is immutable during retrieval
3. Sort results AFTER retrieval (not during)
4. Fixed tile size prevents batch-dependent ordering

□

10.5 Verification Tools

Three tests verify batch-invariance:

1. **Round-Trip Test**: $\text{hash}(\text{compress}(\text{expand}(\text{compress}(M)))) = \text{hash}(\text{compress}(M))$
2. **Determinism Test**: 100 trials produce exactly 1 unique hash
3. **Batch Invariance Test**: Tile sizes $[1, 8, 32, 128, 1024]$ produce identical hashes

10.6 Cross-Instance Relationships

Instances can reference other instances for complex cognition:

- **supersedes**: This memory replaces another
- **derivedFrom**: Synthesized from multiple observations
- **contradicts**: Conflicts with another (triggers resolution)
- **supports**: Corroborates another (boosts confidence)

10.7 Compression Profiles

Four profiles optimize for different use cases:

Table 10: Compression Profiles

Profile	**Target**	**Max Tokens**
context_injection	LLM context window	100
persistent_storage	Disk, version control	Unlimited
api_transport	Network transmission	1000
archive	Rarely accessed	Deferred

10.8 Performance Budget

Table 11: Performance Targets

Operation	**Target Latency**
Compression (100 memories)	< 10ms
Expansion (100 instances)	< 5ms
Exact retrieval	< 2ms
Semantic retrieval	< 50ms
Round-trip verification	< 50ms
Determinism test (100 trials)	< 500ms

11 Falsifiability Criteria

The USD Cognitive Substrate thesis would be **FALSIFIED** if:

1. **Composition Failure**: LIVRPS produces paradoxes in >1% of configurations
2. **Learning Instability**: Mycelium weights oscillate or degenerate
3. **Safety Floor Violation**: Any execution path allows weights below floors
4. **Determinism Failure**: With ThinkingMachines, different outputs in >0.01% of cases
5. **Practical Inferiority**: Simpler system achieves equivalent accuracy
6. **BCM Batch-Invariance Failure**: Trail state affects routing
7. **Grounding Non-Determinism**: Oracle queries produce different results
8. **Tile Size Variance (v7.1.0)**: Different tile sizes produce different results
9. **Aggregation Non-Determinism (v7.1.0)**: Same instances produce different aggregated confidence
10. **Retrieval Variance (v7.1.0)**: Same query on same bank produces different results

Current Status: None observed. All 730 grounded queries bit-identical. All 1,047 tests passing. Batch invariance verified across tile sizes [1, 8, 32, 128, 1024].

12 Conclusion

The USD Cognitive Substrate demonstrates that USD composition semantics—designed for visual effects pipeline conflict resolution—are equally applicable to cognitive state management in LLM applications.

Version 7.0.0 extends the architecture with:

- **Grounding Layer**: ACCESS over LEARN, 730/730 determinism
- **BCM Learning**: Trail-based confidence, batch-invariant
- **Knowledge Prims**: 168,000× speedup for factual retrieval

Version 7.1.0 adds Cognitive Batch Invariance:

- **Fixed Tile Size**: `COGNITIVE_TILE_SIZE=32` for all operations
- **Aggregation Strategies**: Five strategies with Kahan summation
- **Retrieval Invariance**: Same query → same results
- **Cross-Instance Relationships**: Graph structure for complex cognition

When deployed with ThinkingMachines kernels, the system provides a formally verifiable guarantee: **same input + same state → same output**. This transforms LLM applications from probabilistic systems into deterministic functions, enabling reproducibility, testing, auditing, and accountability.

Acknowledgments

The author thanks Pixar Animation Studios for USD, whose composition semantics inspired this architecture, and the ThinkingMachines Lab for batch-invariant inference research He and Thinking Machines Lab [2025].

Code and Data Availability

- **Specification**: https://github.com/JosephOIbrahim/usd-cognitive-substrate
- **Implementation**: https://github.com/JosephOIbrahim/Orchestra (v7.0.0, 1,047 tests)
- **DOI**: 10.5281/zenodo.18332346

References

John R. Anderson. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, 2007.

Stan Franklin, Tamas Madl, Sidney D'Mello, and Javier Snaider. LIDA: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, 6(1):19–41, 2016.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. Thinking Machines Lab: Connectionism, September 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

John E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

Pixar Animation Studios. Universal scene description. Open Source Project, 2016. URL https://graphics.pixar.com/usd/.