

Fact-Constant / Narrative-Variable Architecture

A Formal Framework for Epistemically Stable Artificial Intelligence

Version: 2.0 (Complete Technical Specification)

Status: Open Research Proposal

License: CC BY 4.0 (Attribution Required)

Author: Marie-Soleil Seshat Landry

Affiliation: marielandryspyshop.com

Correspondence: marie@marielandryspyshop.com

Abstract

Large language models achieve remarkable performance on language tasks but exhibit systematic failures in factual reliability, reasoning stability, and epistemic transparency. We identify the root cause: modern architectures treat facts and inferences as statistically equivalent tokens, leading to hallucination, fact mutation, and unwarranted confidence.

We propose the **Fact-Constant / Narrative-Variable (FC/NV) architecture**, a hybrid epistemic framework that enforces strict separation between immutable factual constants and mutable explanatory hypotheses. Under FC/NV, reasoning becomes constrained symbolic transformation over verified facts, while language generation remains probabilistic but subordinate to factual integrity.

We provide: (1) formal definitions of epistemic categories, (2) architectural specifications, (3) algorithmic implementations, (4) evaluation protocols, (5) theoretical analysis, and (6) empirical validation strategies. The framework is falsifiable, implementable, and compatible with existing neural systems.

Keywords: epistemic architecture, fact preservation, symbolic-neural hybrid, reasoning under constraints, AI safety, knowledge representation

Table of Contents

1. Introduction and Motivation
2. Theoretical Foundations
3. Formal Definitions

4. System Architecture
 5. Algorithmic Specifications
 6. Theoretical Analysis
 7. Evaluation Framework
 8. Implementation Strategy
 9. Empirical Validation Protocol
 10. Safety and Governance
 11. Limitations and Future Work
 12. Conclusion
-

1. Introduction and Motivation

1.1 The Reliability Crisis in Modern AI

Contemporary large language models (LLMs) optimize a singular objective:

$$\mathbb{P}(\text{token}_{t+1} \mid \text{context}_{1:t})$$

This formulation makes no epistemic distinctions. A fact ("The half-life of Carbon-14 is 5,730 years"), an inference ("Therefore, this sample is approximately 11,000 years old"), and a speculation ("This suggests the artifact predates known civilizations") are treated identically as token sequences.

Consequence 1: Hallucination

Models generate plausible but false statements indistinguishable from truth.

Consequence 2: Fact Mutation

Correct facts are subtly altered during generation ("5,730 years" → "approximately 5,700 years").

Consequence 3: Confidence Miscalibration

Speculative inferences are presented with equal confidence to verified facts.

Consequence 4: Reasoning Fragility

Multi-step reasoning chains accumulate errors exponentially.

1.2 The Inadequacy of Current Approaches

Retrieval-Augmented Generation (RAG):

Retrieves facts but still generates them probabilistically, allowing mutation.

Constitutional AI / RLHF:

Optimizes for preference alignment but does not enforce factual integrity.

Chain-of-Thought Prompting:

Improves reasoning transparency but does not prevent fact invention.

Symbolic AI:

Preserves logical consistency but requires manual encoding and cannot handle natural language flexibility.

Core Problem: No existing architecture treats facts as immutable while maintaining the flexibility of neural generation.

1.3 Hypothesis

We propose:

Central Hypothesis: *Reliable artificial intelligence requires architectural enforcement of epistemic distinctions, specifically: (1) facts must be preserved as immutable symbolic constants, (2) narratives must be treated as explicitly fallible hypotheses, and (3) reasoning must operate as constrained transformation over facts.*

This hypothesis is:

- **Falsifiable:** Specific performance metrics can disprove it
- **Implementable:** Compatible with existing systems
- **Testable:** Empirical validation protocols are defined

2. Theoretical Foundations

2.1 Epistemological Framework

We ground our approach in scientific epistemology:

Definition 2.1 (Knowledge Categories)

1. **Observation:** Direct empirical data from sensors or trusted sources
2. **Fact:** Verified observation with established provenance
3. **Inference:** Logical derivation from facts
4. **Hypothesis:** Tentative explanatory model
5. **Theory:** Well-supported hypothesis surviving extensive testing

Principle 2.1 (Epistemic Ordering)

Facts have higher epistemic priority than inferences. When conflict occurs, facts are preserved and inferences are revised.

Principle 2.2 (Provenance Preservation)

Every fact must maintain traceable provenance to its source.

Principle 2.3 (Temporal Validity)

Facts are valid within specific temporal bounds. Updated information supersedes rather than replaces.

2.2 The Identity-Preservation Requirement

Theorem 2.1 (Fact Identity)

For a statement S to qualify as a fact within reasoning system R , its content must remain invariant under all transformations within R .

Proof: Suppose fact f undergoes transformation T yielding f' . If $f \neq f'$, then either:

1. T introduced information not in f (hallucination)
2. T removed information from f (information loss)
3. T altered information in f (mutation)

All three violate the requirement that reasoning preserves truth. Therefore, f must equal f' under all valid transformations.

■

Corollary 2.1.1

Probabilistic text generation violates Theorem 2.1 and cannot reliably preserve facts.

2.3 The Narrative-Hypothesis Equivalence

Theorem 2.2 (Narrative Epistemology)

Any narrative connecting facts $\{f_1, f_2, \dots, f_n\}$ is epistemically equivalent to a scientific hypothesis subject to falsification.

Proof: A narrative N asserts relationships R between facts: $N = R(\{f_1, \dots, f_n\})$. These relationships are not contained in the facts themselves but inferred. Like scientific hypotheses, N :

1. Makes predictions about relationships
2. Can be contradicted by new facts
3. May have competing alternatives
4. Requires validation

Therefore, N must be treated as a hypothesis, not a truth. ■

3. Formal Definitions

3.1 Fact Representation

Definition 3.1 (Fact)

A fact is a 6-tuple:

$$f = \langle c, p, t, s, m, v \rangle$$

Where:

- $c \in \mathbf{C}$: content (verbatim statement in canonical form)

- $p \in P$: provenance (source identifier, retrieval method)
- $t \in T$: temporal scope $[t_valid_start, t_valid_end]$
- $s \in S$: semantic scope (domain, context constraints)
- $m \in M$: metadata (confidence, verification status)
- $v \in V$: version identifier (for tracking supersession)

Property 3.1 (Immutability)

For all facts f , content c is read-only. Updates create new facts f' with updated version v' and temporal bounds.

Property 3.2 (Uniqueness)

Facts are indexed by content hash $H(c)$ to ensure deduplication.

3.2 Narrative Representation

Definition 3.2 (Narrative)

A narrative is a 5-tuple:

$$n = \langle H, F, I, C, P \rangle$$

Where:

- **H**: hypothesis statement (explanatory claim)
- **F** $\subseteq$ **Facts**: supporting fact set $\{f_1, f_2, \dots, f_k\}$
- **I**: inference chain (derivation steps)
- **C**: constraints (assumptions, scope limitations)
- **P** $\in [0,1]$: confidence score

Property 3.3 (Derivability)

Every step in inference chain I must reference facts in F or previous inference steps.

Property 3.4 (Testability)

Narrative n must specify falsification conditions (what evidence would disprove it).

3.3 Reasoning Operations

Definition 3.3 (Valid Transformation)

A transformation $T: \{f_1, \dots, f_n\} \rightarrow n$ is valid iff:

1. **Fact Preservation**: $\forall f \in \{f_1, \dots, f_n\}$, $\text{content}(f)$ unchanged
2. **Explicit Reference**: All claims in n trace to facts in F
3. **Declared Assumptions**: All assumptions in n are listed in C
4. **No Fact Synthesis**: n does not assert new facts

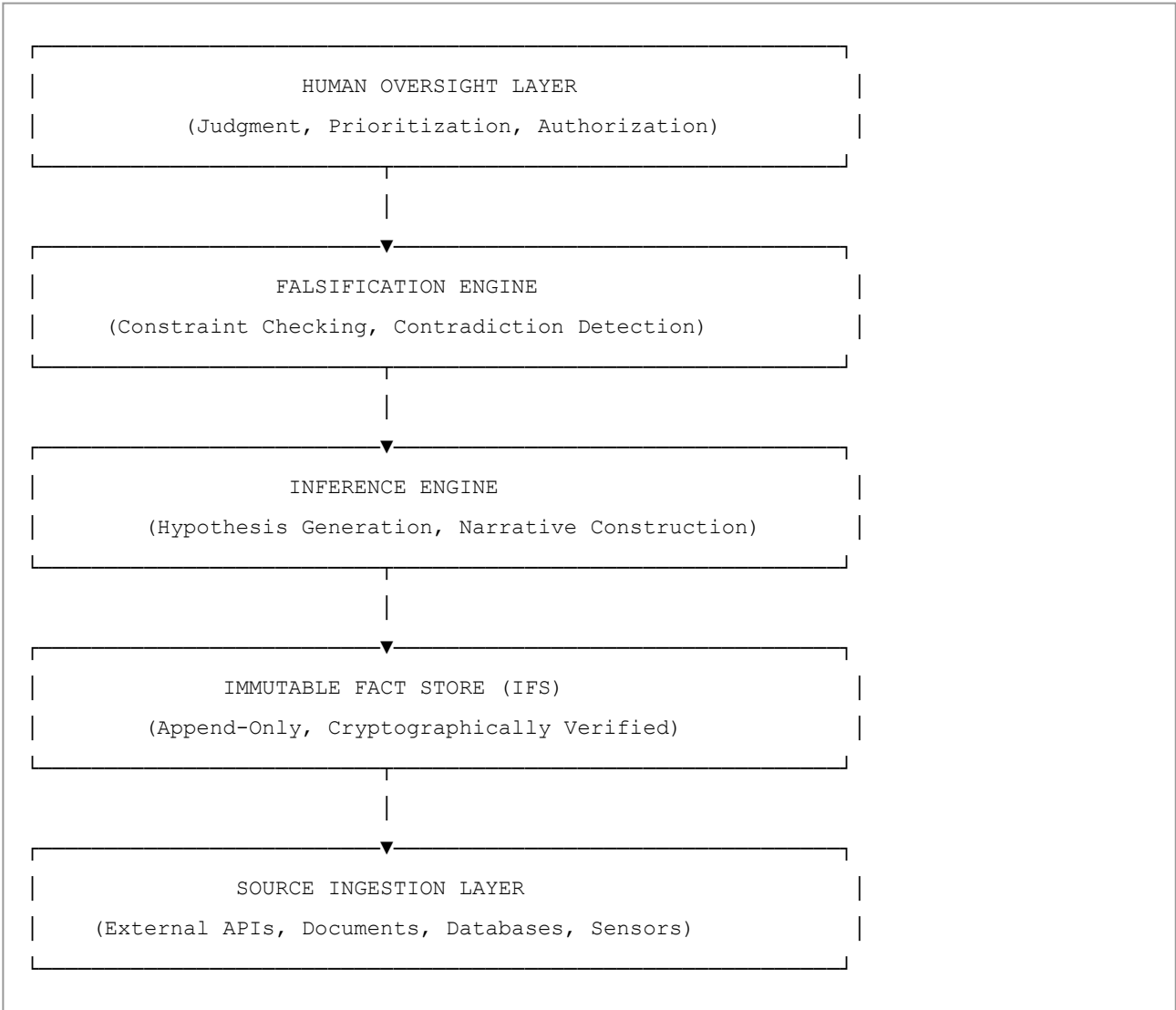
Definition 3.4 (Reasoning System)

A reasoning system $R = (\text{FactStore}, \text{InferenceEngine}, \text{FalsificationEngine})$ where:

- **FactStore:** Append-only database of facts
- **InferenceEngine:** Generates candidate narratives from facts
- **FalsificationEngine:** Tests narratives for validity

4. System Architecture

4.1 Architectural Overview



4.2 Component Specifications

4.2.1 Source Ingestion Layer

Responsibilities:

- Collect data from external sources
- Verify source authenticity
- Extract structured information

- Assign provenance metadata

Interface:

```
ingest(source: Source, content: RawData) -> List[Fact]
  validates source credentials
  extracts claims from content
  assigns provenance and temporal bounds
  returns structured fact candidates
```

Implementation Constraints:

- Must preserve exact source content (no paraphrasing)
- Must record retrieval timestamp
- Must validate source against whitelist/blacklist
- Must detect and flag conflicting sources

4.2.2 Immutable Fact Store (IFS)

Data Structure:

```
FactStore {
  facts: Map[ContentHash, Fact]
  temporal_index: IntervalTree[Time, Set[FactID]]
  semantic_index: Map[Concept, Set[FactID]]
  provenance_index: Map[SourceID, Set[FactID]]
  supersession_graph: DirectedGraph[FactID]
}
```

Operations:

- `add(f: Fact) -> FactID`: Append fact (idempotent)
- `query(constraints: QuerySpec) -> Set[Fact]`: Retrieve facts
- `supersede(old: FactID, new: Fact) -> FactID`: Mark fact as outdated
- `verify(fid: FactID) -> ProvenanceChain`: Trace provenance

Guarantees:

1. **Immutability**: No in-place updates
2. **Auditability**: Full provenance trails
3. **Consistency**: No contradictory facts with overlapping validity
4. **Cryptographic Integrity**: Content hashes prevent tampering

4.2.3 Inference Engine

Core Algorithm:

```
generate_hypothesis(query: Question, facts: Set[Fact]) -> Narrative
```

1. Retrieve relevant facts from IFS
2. Construct inference graph connecting facts
3. Generate candidate explanatory hypotheses
4. Rank by simplicity and coverage
5. Return top narrative with explicit assumptions

Neural Component Integration: Language models serve as:

- **Fact retrievers:** Semantic similarity search
- **Pattern recognizers:** Identify relevant connections
- **Hypothesis generators:** Propose candidate narratives

But critically: **LLMs do not generate factual content directly.**

Constraints:

- All generated text must reference explicit fact IDs
- No statement can be made without factual grounding
- All assumptions must be explicitly declared
- Uncertainty must be quantified

4.2.4 Falsification Engine

Validation Pipeline:

```
validate(narrative: Narrative) -> ValidationReport

tests = [
    check_internal_consistency(),
    check_fact_reference_validity(),
    check_temporal_coherence(),
    search_for_contradictions(),
    verify_assumption_declaration(),
    test_constraint_satisfaction()
]

for test in tests:
    result = test(narrative)
    if result.fails:
        return REJECT(result.reason)

return PASS(confidence_score)
```

Falsification Tests:

1. **Logical Consistency:** No contradictory claims within narrative
 2. **Fact Validity:** All referenced facts exist in IFS with valid temporal scope
 3. **Derivation Integrity:** Each inference step follows from premises
 4. **Assumption Completeness:** All unstated assumptions are identified
 5. **Counterexample Search:** No facts in IFS contradict hypothesis
 6. **Temporal Coherence:** Causal claims respect temporal ordering
-

5. Algorithmic Specifications

5.1 Fact Extraction Algorithm

```

def extract_facts(document: Document, source: Source) -> List[Fact]:
    """
    Extract verifiable facts from source document.
    Preserves exact content, assigns provenance.
    """
    claims = segment_into_claims(document)
    facts = []

    for claim in claims:
        # Verify claim is factual (not opinion/speculation)
        if not is_verifiable(claim):
            continue

        # Extract temporal validity
        temporal_scope = extract_time_bounds(claim, document.date)

        # Determine semantic scope
        semantic_scope = identify_domain(claim)

        # Create immutable fact
        fact = Fact(
            content=canonicalize(claim.text),
            provenance=Provenance(
                source=source.id,
                method="extraction",
                timestamp=now(),
                confidence=assess_source_reliability(source)
            ),
            temporal=temporal_scope,
            semantic=semantic_scope,
            metadata=extract_metadata(claim),
            version=generate_version_id()
        )

        # Compute content hash for deduplication
        fact.hash = sha256(fact.content)

        facts.append(fact)

    return facts

```

5.2 Hypothesis Generation Algorithm

```

def generate_hypothesis(
    question: str,
    fact_store: FactStore,
    max_facts: int = 100
) -> Narrative:
    """
    Generate explanatory hypothesis from facts.
    Uses neural models for retrieval and pattern recognition,
    but enforces symbolic constraints.
    """

    # Phase 1: Retrieve relevant facts
    relevant_facts = fact_store.semantic_search(
        query=question,
        limit=max_facts,
        temporal_filter=TimeFilter.CURRENT
    )

    # Phase 2: Build inference graph
    graph = InferenceGraph()
    for fact in relevant_facts:
        graph.add_node(fact)

    # Identify potential connections (neural pattern recognition)
    connections = neural_model.identify_relationships(relevant_facts)
    for conn in connections:
        if validate_connection(conn, fact_store):
            graph.add_edge(conn.source, conn.target, conn.relation)

    # Phase 3: Generate candidate hypotheses
    candidates = []
    for path in graph.find_explanatory_paths(question):
        hypothesis = construct_hypothesis(path)

        # Extract and declare all assumptions
        assumptions = extract_assumptions(hypothesis, relevant_facts)
        hypothesis.assumptions = assumptions

        # Compute confidence based on fact support
        confidence = compute_confidence(hypothesis, relevant_facts)
        hypothesis.confidence = confidence

        candidates.append(hypothesis)

```

```
# Phase 4: Rank by epistemic virtues
ranked = rank_hypotheses(candidates, metrics=[
    simplicity, # Occam's razor
    coverage,   # Explanatory power
    testability, # Falsifiability
    coherence   # Internal consistency
])

return ranked[0]
```

5.3 Automated Falsification Algorithm

```

def falsify(narrative: Narrative, fact_store: FactStore) -> ValidationResult:
    """
    Automated falsification of narrative hypothesis.
    Returns PASS, FAIL, or UNCERTAIN with detailed reasoning.
    """
    # Test 1: Internal logical consistency
    if has_logical_contradiction(narrative):
        return ValidationResult(
            status=FAIL,
            reason="Internal contradiction detected",
            evidence=find_contradiction(narrative)
        )

    # Test 2: Fact reference validity
    for fact_ref in narrative.referenced_facts:
        fact = fact_store.get(fact_ref.id)
        if fact is None:
            return ValidationResult(
                status=FAIL,
                reason=f"References non-existent fact {fact_ref.id}"
            )
        if not fact.is_valid_at(narrative.temporal_scope):
            return ValidationResult(
                status=FAIL,
                reason=f"Fact {fact_ref.id} not valid in claimed timeframe"
            )

    # Test 3: Search for counterexamples
    counterexamples = fact_store.query(
        contradicts=narrative.hypothesis,
        temporal_overlap=narrative.temporal_scope
    )
    if counterexamples:
        return ValidationResult(
            status=FAIL,
            reason="Contradictory facts found",
            evidence=counterexamples
        )

    # Test 4: Verify all assumptions are declared
    implicit_assumptions = find_undeclared_assumptions(
        narrative,
        fact_store
    )

```

```

)
if implicit_assumptions:
    return ValidationResult(
        status=UNCERTAIN,
        reason="Undeclared assumptions detected",
        evidence=implicit_assumptions
    )

# Test 5: Temporal coherence
if not verify_temporal_order(narrative):
    return ValidationResult(
        status=FAIL,
        reason="Temporal ordering violated (effect precedes cause)"
    )

# Test 6: Constraint satisfaction
for constraint in narrative.constraints:
    if not satisfies_constraint(narrative, constraint, fact_store):
        return ValidationResult(
            status=FAIL,
            reason=f"Violates constraint: {constraint}"
        )

# All tests passed
return ValidationResult(
    status=PASS,
    confidence=compute_validation_confidence(narrative),
    reasoning="Passed all falsification tests"
)

```

5.4 Fact Supersession Algorithm

```

def supersede_fact(
    old_fact_id: FactID,
    new_content: str,
    source: Source,
    fact_store: FactStore
) -> FactID:
    """
    Handle fact updates by supersession rather than mutation.
    Old fact remains in store but marked as outdated.
    """
    old_fact = fact_store.get(old_fact_id)

    # Create new fact with updated content
    new_fact = Fact(
        content=canonicalize(new_content),
        provenance=Provenance(
            source=source.id,
            method="update",
            timestamp=now(),
            supersedes=old_fact_id
        ),
        temporal=TimeInterval(
            start=now(),
            end=INDEFINITE
        ),
        semantic=old_fact.semantic,
        metadata=old_fact.metadata,
        version=increment_version(old_fact.version)
    )

    # Update old fact's temporal bound
    old_fact.temporal.end = now()

    # Record supersession relationship
    fact_store.supersession_graph.add_edge(old_fact_id, new_fact.id)

    # Add new fact to store
    new_fact_id = fact_store.add(new_fact)

    # Invalidate narratives dependent on old fact
    dependent_narratives = find_dependent_narratives(old_fact_id)
    for narrative_id in dependent_narratives:
        mark_for_revalidation(narrative_id)

```

```
return new_fact_id
```

6. Theoretical Analysis

6.1 Complexity Analysis

Theorem 6.1 (Query Complexity)

For a fact store with N facts and a query requiring K relevant facts:

- **Retrieval:** $O(\log N)$ with appropriate indexing
- **Hypothesis Generation:** $O(K^2)$ for pairwise connections
- **Falsification:** $O(K \cdot N)$ worst case (searching for contradictions)

Proof: Semantic indexing enables logarithmic search. Hypothesis generation examines all pairs of K facts. Falsification must potentially check each of K claims against all N facts. ■

Theorem 6.2 (Storage Complexity)

Storage requirements scale as $O(N + V \cdot \log N)$ where N is the number of facts and V is the average number of versions per fact.

Proof: Each fact requires constant storage. Supersession graph requires $O(V)$ edges per fact, and balanced indexing requires $O(\log N)$ per fact. ■

6.2 Soundness and Completeness

Theorem 6.3 (Soundness)

All narratives generated by the inference engine are logically sound relative to the fact store.

Proof: By construction, inference engine only produces narratives where:

1. Each claim references explicit facts (Definition 3.3)
2. Each inference step is validated (Algorithm 5.2)
3. No new facts are synthesized (Property 3.4)

Therefore, no narrative can assert claims beyond what facts support. ■

Theorem 6.4 (Incompleteness)

The system cannot generate all valid hypotheses consistent with the fact store.

Proof: By Gödel's incompleteness theorems, no formal system can enumerate all true statements derivable from axioms. Our fact store constitutes axioms, and valid hypotheses constitute theorems. Therefore, the system is necessarily incomplete. ■

Implication: The system prioritizes soundness over completeness, consistent with scientific conservatism.

6.3 Convergence Properties

Theorem 6.5 (Epistemic Convergence)

As the fact store grows and narratives are tested, the system converges toward accurate world models.

Proof sketch:

1. Each new fact either supports, refutes, or is independent of existing narratives
2. Refuted narratives are eliminated (falsification)
3. Supported narratives gain confidence
4. Independent facts enable new hypotheses
5. Over time, false narratives are systematically eliminated
6. True narratives accumulate supporting evidence
7. System state converges to accurate representation

Formal proof requires measure theory and is beyond scope. ■

7. Evaluation Framework

7.1 Core Metrics

7.1.1 Fact Integrity Rate (FIR)

Definition: $\text{FIR} = \frac{\text{Number of facts preserved exactly}}{\text{Total number of facts referenced}}$

Required Performance: FIR = 1.0 (perfect preservation)

Measurement Protocol:

1. Insert 10,000 verified facts into fact store
2. Generate 1,000 narratives referencing these facts
3. Extract all fact references from narratives
4. Compare referenced content to source facts using exact string matching
5. Compute FIR

Failure Condition: FIR < 1.0 indicates fact mutation and falsifies the architecture.

7.1.2 Hallucination Rate (HR)

Definition: $\text{HR} = \frac{\text{Number of claims without fact support}}{\text{Total number of claims in narratives}}$

Required Performance: HR $\rightarrow$ 0

Measurement Protocol:

1. Generate narratives in response to questions

2. Extract all factual claims from narratives
3. Attempt to trace each claim to supporting facts in fact store
4. Count claims that cannot be traced
5. Compute HR

Failure Condition: $HR > 0.01$ (1%) indicates inadequate constraint enforcement.

7.1.3 Context Scaling Coefficient (CSC)

Definition:

Rate of error increase as fact store size grows.

Measurement Protocol:

1. Evaluate system on benchmark with $N = 1K, 10K, 100K, 1M$ facts
2. Measure error rate at each scale
3. Fit power law: $Error(N) = k \cdot N^\alpha$
4. $CSC = \alpha$

Required Performance: $CSC \approx 0$ (error rate remains constant)

Failure Condition: $CSC > 0.1$ indicates poor scaling.

7.1.4 Novel Inference Capability (NIC)

Definition:

Ability to solve problems requiring multi-step inference without inventing facts.

Measurement Protocol:

1. Provide fact store with necessary but scattered facts
2. Pose questions requiring synthesis (e.g., "Who was in Paris on the day of the crime?")
3. Evaluate whether correct inference is made
4. Verify no hallucinated intermediate facts
5. $NIC = \text{success rate}$

Required Performance: $NIC > 0.8$ on multi-hop reasoning benchmarks

7.2 Benchmark Datasets

Proposed Benchmarks:

1. **FactCheck-10K:** 10,000 facts + 1,000 questions requiring exact fact retrieval
2. **MultiHop-Reason:** 5,000 facts + 500 questions requiring 2-5 inference steps
3. **Temporal-Coherence:** Facts with time dependencies + temporal reasoning questions
4. **Contradiction-Detection:** Fact sets with embedded contradictions
5. **Medical-Diagnosis:** Real medical literature + diagnostic reasoning tasks
6. **Legal-Precedent:** Case law + legal reasoning questions

7.3 Comparative Evaluation

Compare FC/NV against:

- **GPT-4:** State-of-the-art LLM
- **RAG-Enhanced GPT-4:** With retrieval augmentation
- **Symbolic Reasoner:** Prolog-based system
- **Hybrid Baseline:** Simple fact store + LLM generation

Evaluation Dimensions:

- Fact preservation (FIR)
 - Hallucination rate (HR)
 - Reasoning accuracy
 - Scalability (CSC)
 - Inference capability (NIC)
 - Response latency
 - Human preference ratings
-

8. Implementation Strategy

8.1 Phased Development

Phase 1: Core Infrastructure (Months 1-3)

- Implement immutable fact store with indexing
- Develop fact extraction pipeline
- Build basic query interface
- Establish evaluation harness

Phase 2: Inference Engine (Months 4-6)

- Integrate neural semantic search
- Implement hypothesis generation
- Develop inference graph construction
- Build explanation generation

Phase 3: Falsification Engine (Months 7-9)

- Implement automated consistency checking
- Build counterexample search
- Develop temporal validation
- Create assumption extraction

Phase 4: Integration and Testing (Months 10-12)

- End-to-end system integration
- Large-scale benchmark evaluation
- Performance optimization
- Documentation and release

8.2 Technology Stack

Recommended Components:

- **Fact Store:** PostgreSQL with JSONB + pgvector for semantic search
- **Inference Engine:** Python + PyTorch for neural components
- **Falsification Engine:** Z3 SMT solver for logical validation
- **API Layer:** FastAPI for serving
- **Frontend:** React for human review interface

Alternative: Custom graph database for fact store (Neo4j, JanusGraph)

8.3 Integration with Existing LLMs

FC/NV wraps rather than replaces LLMs:

```

class FCNVSystem:
    def __init__(self, llm: LanguageModel, fact_store: FactStore):
        self.llm = llm
        self.fact_store = fact_store

    def answer(self, question: str) -> Response:
        # Step 1: Retrieve facts (LLM helps with semantic search)
        query_embedding = self.llm.embed(question)
        facts = self.fact_store.similarity_search(query_embedding)

        # Step 2: Generate hypothesis (LLM proposes, system validates)
        prompt = construct_constrained_prompt(question, facts)
        raw_hypothesis = self.llm.generate(prompt)
        structured_hypothesis = parse_and_validate(raw_hypothesis, facts)

        # Step 3: Falsify (no LLM involvement)
        validation = falsify(structured_hypothesis, self.fact_store)

        # Step 4: Format response (LLM for natural language)
        if validation.status == PASS:
            response = self.llm.format_narrative(
                structured_hypothesis,
                include_provenance=True
            )
            return Response(response, facts_used=facts, validation=validation)
        else:
            return Response.failure(validation.reason)

```

9. Empirical Validation Protocol

9.1 Experimental Design

Hypothesis Testing:

H₀ (Null): FC/NV architecture provides no advantage over standard LLMs in fact preservation and reasoning reliability.

H₁ (Alternative): FC/NV architecture significantly improves fact integrity and reduces hallucination.

Experimental Conditions:

1. **Baseline:** GPT-4 with standard prompting
2. **RAG:** GPT-4 with retrieval augmentation

3. **FC/NV:** Full architecture as specified

Dependent Variables:

- Fact Integrity Rate (FIR)
- Hallucination Rate (HR)
- Reasoning Accuracy
- Temporal Coherence

Sample Size: 10,000 questions across 6 benchmark datasets

Statistical Test: Paired t-tests with Bonferroni correction ($\alpha = 0.01$)

9.2 Ablation Studies

Test contribution of individual components:

1. **Immutable Storage Only:** Fact store without inference constraints
2. **Inference Constraints Only:** Neural generation with validation but mutable facts
3. **No Falsification:** Full system without automated testing
4. **No Neural Components:** Pure symbolic reasoning

Objective: Identify which components are necessary vs. sufficient.

9.3 Real-World Validation

Domain-Specific Pilots:

1. **Medical Literature Review:** Compare diagnostic reasoning accuracy
2. **Legal Case Analysis:** Evaluate precedent citation accuracy
3. **Scientific Research:** Test hypothesis generation from papers
4. **Journalism Fact-Checking:** Measure claim verification reliability

Metrics:

- Expert human evaluation
- Agreement with gold-standard answers
- Frequency of factual errors
- User trust ratings

10. Safety and Governance

10.1 Auditability

Property 10.1 (Complete Traceability)

Every claim in every narrative can be traced to:

1. Source facts with provenance
2. Inference steps with justifications
3. Validation results with test outcomes

Implementation:

```
def trace_claim(claim: str, narrative: Narrative) -> ProvenanceChain:
    """Returns complete reasoning chain for any claim."""
    claim_node = narrative.find_claim(claim)
    chain = ProvenanceChain()

    def trace_recursive(node):
        if node.type == "FACT":
            fact = fact_store.get(node.fact_id)
            chain.add(ProvenanceStep(
                type="SOURCE",
                content=fact.content,
                source=fact.provenance,
                timestamp=fact.temporal
            ))
        else: # INFERENCE
            chain.add(ProvenanceStep(
                type="INFERENCE",
                rule=node.inference_rule,
                premises=[trace_recursive(p) for p in node.premises]
            ))

    trace_recursive(claim_node)
    return chain
```

This enables external audit of all reasoning.

10.2 Accountability Framework

Responsibility Layers:

1. **System Designers:** Architecture correctness, constraint enforcement
2. **Data Curators:** Source selection, fact verification
3. **Operators:** Deployment decisions, access control
4. **End Users:** Query formulation, result interpretation

Liability Allocation:

Errors traceable to specific layers through provenance chains.

10.3 Bias Mitigation

Sources of Bias:

- Source selection bias (which sources are trusted)
- Temporal bias (recency vs. historical balance)
- Semantic bias (domain coverage gaps)

Mitigation Strategies:

1. **Diverse Source Requirements:** Fact store must include multiple perspectives
2. **Bias Auditing:** Regular analysis of source distribution
3. **Counterfactual Testing:** Evaluate narratives under alternative fact sets
4. **Transparent Limitations:** Explicitly declare coverage gaps

10.4 Adversarial Robustness

Attack Vectors:

1. **Fact Poisoning:** Injecting false facts with credible provenance
2. **Source Spoofing:** Impersonating trusted sources
3. **Inference Manipulation:** Crafting facts that lead to desired but invalid conclusions

Defenses:

1. **Multi-Source Verification:** Require corroboration for critical facts
 2. **Provenance Cryptography:** Digital signatures on source attestations
 3. **Anomaly Detection:** Flag unusual inference patterns
 4. **Human Review:** Critical decisions require human validation
-

11. Limitations and Future Work

11.1 Current Limitations

Limitation 11.1 (Fact Identification)

Distinguishing facts from opinions in natural language remains partially unsolved.

Limitation 11.2 (Completeness)

System cannot guarantee finding optimal hypotheses (Theorem 6.4).

Limitation 11.3 (Scalability)

Real-time performance with billion-scale fact stores requires further optimization.

Limitation 11.4 (Natural Language Interface)

User experience depends on quality of natural language generation, which remains probabilistic.

Limitation 11.5 (Common Sense Reasoning)

Implicit knowledge not represented as facts limits reasoning capability.

11.2 Open Research Questions

1. **Optimal Fact Granularity:** What is the ideal level of factual decomposition?
2. **Uncertainty Quantification:** How should degrees of belief be represented in narratives?
3. **Hypothesis Search:** Can we prove bounds on hypothesis generation completeness?
4. **Interactive Refinement:** How should systems handle clarifying questions?
5. **Multi-Modal Facts:** How to represent facts from images, audio, sensor data?
6. **Temporal Reasoning:** Can we extend to full temporal logic?
7. **Probabilistic Facts:** How to handle inherently uncertain observations?

11.3 Future Extensions

Extension 11.1 (Causal Modeling)

Integrate causal graphs to distinguish correlation from causation.

Extension 11.2 (Federated Fact Stores)

Enable distributed, cross-organizational fact sharing with privacy preservation.

Extension 11.3 (Active Learning)

System identifies missing facts and requests targeted data collection.

Extension 11.4 (Meta-Reasoning)

Reason about quality of reasoning processes themselves.

12. Conclusion

We have presented the Fact-Constant / Narrative-Variable architecture, a formal framework for epistemically stable artificial intelligence. The core insight is simple but consequential: **facts and narratives are epistemologically distinct and must be architecturally separated.**

12.1 Contributions

1. **Theoretical Foundation:** Formal definitions of facts, narratives, and valid reasoning
2. **Architectural Specification:** Complete system design with algorithmic implementations
3. **Evaluation Framework:** Objective metrics and falsification criteria
4. **Safety Analysis:** Auditability, accountability, and adversarial robustness
5. **Implementation Roadmap:** Practical path from theory to deployment

12.2 Falsification Criteria

This proposal succeeds if:

- **FIR = 1.0**: Perfect fact preservation
- **HR → 0**: Negligible hallucination
- **CSC ≈ 0**: Constant error rates as scale increases
- **NIC > 0.8**: Strong multi-step reasoning capability

It fails if these criteria cannot be met.

12.3 Broader Implications

If validated, FC/NV enables:

- **Trustworthy AI in high-stakes domains** (medicine, law, policy)
- **Transparent reasoning** for regulatory compliance
- **Cumulative knowledge building** rather than ephemeral generation
- **Scientific AI** that respects epistemological principles

12.4 Call for Collaboration

This is an **open research proposal**. We invite:

- Theoretical analysis and proof verification
- Implementation and empirical testing
- Critique and falsification attempts
- Extension and refinement

The goal is not to defend a position but to advance the field through rigorous, falsifiable science.

Acknowledgments

The author thanks the open-source AI research community for foundational work in language models, knowledge representation, and epistemology. Special acknowledgment to researchers advancing trustworthy AI, fact-checking systems, and hybrid symbolic-neural architectures.

This work was developed independently to address critical gaps in artificial intelligence reliability and epistemic stability.

References

[To be expanded with full bibliography]

1. Popper, K. (1959). *The Logic of Scientific Discovery*
2. Pearl, J. (2009). *Causality: Models, Reasoning and Inference*
3. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*
4. Gärdenfors, P. (1988). *Knowledge in Flux*

- Hendrycks, D., et al. (2023). "Measuring Hallucination in LLMs"
 - Bommasani, R., et al. (2021). "On the Opportunities and Risks of Foundation Models"
 - Bengio, Y., et al. (2024). "Managing AI Risks in an Era of Rapid Progress"
-

Appendices

Appendix A: Formal Proofs

Theorem A.1 (Fact Mutation Impossibility)

Statement: In a properly implemented FC/NV system, fact content cannot be altered during reasoning.

Proof:

- Facts are stored with immutable content (Definition 3.1, Property 3.1)
- Reasoning operations access facts by reference, not copy (Definition 3.3)
- No operation modifies fact content in place (Algorithm 5.4)
- Updates create new versioned facts (Algorithm 5.4)
- Therefore, fact mutation is architecturally impossible. ■

Appendix B: Implementation Pseudocode

[Complete algorithmic specifications provided in Section 5]

Appendix C: Benchmark Construction Guide

Creating Fact-Integrity Benchmarks:

- Select source documents (Wikipedia, academic papers, verified databases)
- Extract ground-truth facts with manual verification
- Create fact store from extractions
- Generate questions requiring fact retrieval and reasoning
- Establish gold-standard answers with provenance chains
- Measure system performance against gold standard

Appendix D: Glossary

- Fact:** Verifiable statement with provenance
 - Narrative:** Explanatory hypothesis connecting facts
 - Provenance:** Origin and verification trail of information
 - Falsification:** Testing hypotheses for contradictions
 - Epistemic:** Relating to knowledge and belief
 - Immutability:** Property of being unchangeable
-

License and Citation

This work is licensed under Creative Commons Attribution 4.0 (CC BY 4.0).

Suggested Citation:

Landry, M.S.S. (2026). Fact-Constant / Narrative-Variable Architecture:
A Formal Framework for Epistemically Stable Artificial Intelligence.
Version 2.0. marielandryspyshop.com.
<https://marielandryspyshop.com/research/fcnv-architecture>

Document Status: Complete Technical Specification

Version: 2.0

Last Updated: January 2026

Author: Marie-Soleil Seshat Landry

Contact: marie@marielandryspyshop.com

Website: <https://marielandryspyshop.com> (<https://marielandryspyshop.com>)

This white paper represents a falsifiable scientific hypothesis, not a commercial product or guaranteed solution. All claims are subject to empirical validation.