

Memnora: Identity-Governed Memory and Accountable Action

A Design-Science Dissertation Manuscript on the Trynary
Method, TempleBoard-T1, and MCSPP

Dwight R. Vaughn

October 16, 2026

Architectural synthesis | Formal interpretation | Executed finite analysis | Reproducible research program

Independent scholarship. No university affiliation, degree submission, or institutional approval is represented.

Author's note and research status

This manuscript organizes Dwight R. Vaughn's Memnora work into a dissertation-style argument in computing and systems design. It is an independent research manuscript, not a dissertation submitted to or approved by a university. Its contribution is a documented architectural synthesis, a proposed formal interpretation, an executable abstract gate analysis, and a reproducible evaluation program. It does not report a completed human-subject study or a tested TempleBoard device.

The author supplied the underlying concepts and source documents. AI assistance was used for source organization, literature discovery, drafting, mathematical clarification, code preparation, and document production. The Python model described in Chapter 7 was executed during preparation. Its limited results are distinguished throughout from source-document claims and proposed future experiments. Before academic submission, the author should review every argument and follow the receiving institution's authorship and AI-disclosure policies.

The primary technical sources reviewed are the TempleBoard working specification [P1], the MCSPP v0.1 final specification [P2], and the Memnora Identity proposal developed in this project [P3]. The scope is selected computing work rather than a complete scholarly edition of all the author's books and archives. Revisions are synthesized by subject; their appearance together does not establish a single already-integrated implementation.

Abstract

Persistent AI assistance creates a problem that extends beyond generating useful language. A system must preserve relevant information across sessions, distinguish a remembered claim from verified evidence, protect identity and permissions during change, and connect consequential actions to their authorizing conditions. This manuscript investigates that problem through the Memnora framework and two related designs: TempleBoard-T1 and the Memnora Causal State & Provenance Protocol, MCSPP v0.1.

The research uses design analysis to reconstruct the architecture and operationalize its principal terms. The Trynary Method is interpreted as a decision discipline with three explicit outcomes: Execute (+1), Hold (0), and Reject (-1). Hold preserves unresolved conditions without converting uncertainty into permission. TempleBoard supplies a proposed separation of computation, governance, and action. MCSPP distinguishes state, identity, and lineage, requiring semantic acceptance beyond structural validity. These elements are combined into an architectural thesis: persistent assistance becomes more accountable when memory changes and actions pass explicit identity, provenance, evidence, policy, and audit boundaries.

A finite Python model enumerates all 729 input combinations for six abstract checks. The specified conjunction-based gate produces one Execute combination, 63 Hold combinations, and 665 Reject combinations. A deliberately permissive four-of-six majority comparator authorizes 72 combinations blocked by the stricter gate. These counts describe decision-policy behavior, not real-world error rates or protocol conformance. Conditional proofs and failure analysis identify further needs in atomic commit, time evidence, recovery, and actuator isolation.

The manuscript concludes with experiments for persistent memory, uncertainty handling, provenance reconstruction, and physical action control. It positions the work as an engineering research program whose claims can be strengthened, revised, or rejected by reproducible evidence. It makes no finding about subjective machine consciousness, quantum effects, clinical efficacy, or hardware safety certification.

Keywords: Memnora; Trynary Method; identity continuity; persistent memory; provenance; AI governance; TempleBoard; MCSPP; design science.

Contents

Author's note and research status	2
Abstract	3
1. Introduction: from conversation to accountable continuity	5
2. Sources, method, and related research	7
3. Operational foundations of the Trynary Method	9
4. Memnora's memory, identity, and interpretive architecture	11
5. TempleBoard-T1: translating architecture into control boundaries	13
6. MCSPP: semantic validity and causal lineage	15
7. Formal model and executed finite analysis	17
8. Evaluation design and reproducibility	20
9. Discussion, implications, and limitations	22
10. Conclusion and research agenda	24
Appendix A. Requirements and test catalog	25
Appendix B. Reproducible gate model	27
References and primary-source register	29

1. Introduction: from conversation to accountable continuity

1.1 The problem of continuity

A conversational system can produce an apparently coherent answer while losing the conditions that made an earlier answer valid. A project may change ownership, an instruction may be revoked, a document may be corrected, or an initially plausible assumption may fail. Remembering the earlier text is insufficient if the system cannot determine its present authority and evidential status. The practical research problem is therefore the governance of continuity: what may persist, what may change, and what justifies action after change.

Memnora treats memory and identity as central design concerns. In the reviewed TempleBoard material, information enters through acquisition and normalization stages, becomes structured candidate material, is compared with memory, and passes through trust and identity checks before adjudication [P1]. MCSPP approaches the same concern at the event level: an accepted transition requires more than a well-formed message [P2, sections 1-3]. These sources support a thesis about accountable state change. They do not, by themselves, establish a completed machine or a demonstrated advantage over existing systems.

Consider a research workspace that stores an equipment setting. On Monday a source recommends the setting; on Tuesday another source contradicts it; on Wednesday the operator's authority expires. A system that retrieves the Monday statement may still produce fluent instructions. A governed system must ask separately whether the fact remains supported, whether the requested operation is permitted, and whether execution can be reconstructed afterward. Those questions motivate the three-way decision and the separation of state from identity and lineage.

1.2 Research questions and thesis

The principal question is: how can a persistent AI interface preserve useful continuity while ensuring that unsupported or unauthorized transitions do not become authoritative? Four subsidiary questions guide the analysis. RQ1 asks how Hold can be represented as a meaningful computational outcome. RQ2 asks how memory can change without silently rewriting identity or authority. RQ3 asks how an action can be linked to its actual evidence, permissions, and observed consequences. RQ4 asks what tests distinguish a credible governance mechanism from persuasive descriptions of one.

The thesis is conditional and architectural: if memory mutation and action execution are mediated by explicit gates, if those gates operate on verifiable conditions, and if accepted transitions are committed with reconstructable lineage, then the system can enforce identifiable accountability properties. Whether those properties improve practical performance remains an empirical question. The thesis does not require a new numerical substrate; ordinary binary hardware can implement a three-outcome policy.

This manuscript contributes an operational vocabulary, an integration model, conditional invariants, an executed finite analysis, and an evaluation plan. The proposed novelty lies in the particular synthesis and its implementation discipline. No comprehensive patent search or exhaustive prior-art review has been performed. Consequently, the work makes no claim that three-valued decisions, external memory, provenance, or separated controllers originated with Memnora.

1.3 Boundaries of the investigation

The unit of analysis is an evidence-bearing state transition. A transition may commit a memory, authorize a tool call, accept a protocol event, or permit an actuator command. The same questions recur, but these operations are not interchangeable. A software memory commit has different failure modes from a relay switching under electrical load. The research therefore preserves distinct software, protocol, and physical-control layers.

The term living interface is treated as the author's product and conceptual language. Within the technical argument it denotes continuity through interaction, retained context, revision, and feedback. Observable continuity cannot settle the presence of subjective experience. Likewise, temple-derived organization is considered a source of design hypotheses, not historical evidence that an ancient structure implemented a computer. Keeping these distinctions explicit allows the engineering proposal to be evaluated on its own merits.

2. Sources, method, and related research

2.1 Primary corpus and evidential discipline

TempleBoard [P1] is an extended working document containing architectural mappings, hardware zones, object models, governance services, and proposed tests. Its own engineering-status section distinguishes specified roles from unresolved fabrication details. That distinction is retained here. The architecture is a documented design artifact; exact electrical realization and performance require additional evidence. Some passages are conversational development notes rather than finalized normative requirements.

MCSPP [P2] is more tightly bounded. It labels its v0.1 core frozen and provides event types, relationships, profile choices, validation rules, receipts, and conformance intentions. Freezing a specification establishes a reference for implementation; it does not demonstrate that an implementation passes its tests. The Memnora Identity proposal [P3] supplies a scoped application involving account access and persistent workspaces. Its biometric component remains proposed.

Source selection deliberately excludes similarly named but unrelated materials. A DARS safety-simulation document retrieved during preparation concerns a medical concept, not the computing governance service described in [P1]. No health or therapeutic numbers from that document are used. This exclusion illustrates a general problem for persistent memory: acronym matching and semantic resemblance must not substitute for source and domain verification.

2.2 Method: reconstruction, formalization, and evaluation design

The method has four stages. First, source reconstruction identifies repeated objects, boundaries, and declared requirements. Second, operationalization replaces broad terms with observable variables or explicit predicates. Third, a minimal executable model checks a narrow consequence of those definitions. Fourth, an evaluation program specifies how broader claims would be tested. Findings at one stage do not automatically transfer to another.

The manuscript uses three evidential categories. A documentary finding states what a reviewed specification contains. An analytical result follows from explicitly stated definitions or executed code. An empirical result would require observations from an implemented system, people, or hardware. Only the first two categories are established here. Examples, proposed benchmarks, and target behavior are labeled so that they cannot be mistaken for collected observations.

This is a design-science approach in the ordinary methodological sense: construct and analyze an artifact intended to address a practical problem, then specify how its usefulness can be evaluated. It is not an institutional claim or a claim to have completed an externally reviewed design-science study. Reflexivity is necessary because the manuscript interprets the author's own work; adversarial tests and independent replication are therefore part of the research design.

2.3 Related work and the limits of comparison

Lewis and colleagues combine parametric language-model knowledge with retrieved non-parametric memory in retrieval-augmented generation [R1]. That work provides an important precedent for consulting external information during generation. Memnora's proposed emphasis is narrower at the governance boundary: retained material should carry ownership, status, and authority constraints. This comparison identifies a design concern; it does not show that ordinary retrieval systems lack access controls.

Yao and colleagues' ReAct interleaves reasoning and task actions [R2]. It offers a useful reference for an observe-and-act loop. The extension proposed here is to treat candidate actions as requests evaluated by a separately specified authority boundary. Park and colleagues describe agents with stored experiences, retrieval, reflection, and planning [R3]. Their work shows that persistent agent memory is an established research area. Behavioral believability, however, is a different evaluation objective from permission enforcement or audit reconstruction.

W3C PROV-DM models provenance through entities, activities, agents, and relations [R4]. It provides a comparison vocabulary for MCSPP's states, events, and responsible identities. NIST AI RMF organizes risk-management activities through Govern, Map, Measure, and Manage [R5]. It supplies a broader organizational context, not certification of this architecture. WebAuthn describes public-key authentication with authenticators [R6]; it helps separate user verification from centrally processing facial images.

The literature reviewed here is selective. It is sufficient to locate the proposal among external-memory agents, provenance systems, authentication, and AI governance, but insufficient to support an exhaustive originality claim. The manuscript's synthesis and formal interpretation are its own analysis. Literature findings support context rather than validating Memnora by association.

3. Operational foundations of the Trynary Method

3.1 Three outcomes and two different vocabularies

The author uses the Trynary Method to organize reception, evaluation, and release. The TempleBoard technical model also assigns +1 to Execute, 0 to Hold, and -1 to Reject [P1, Trinary State Model]. A workflow stage and a decision outcome are different types of object. Receiving input is not equivalent to accepting an action, and releasing an answer is not equivalent to rejecting it. This manuscript keeps the three-stage interpretive cycle separate from the three-outcome gate.

Let $D = \{-1, 0, +1\}$ be the decision set. Execute permits a specified transition under the active policy. Hold preserves the request without executing it because relevant conditions remain unresolved. Reject denies the present request because at least one required condition has failed. These are decisions about an operation, not truth values for every statement in a conversation. A fact can remain uncertain while a low-risk retrieval operation is permitted.

The implementation may encode Hold as 00, Execute as 01, Reject as 10, and Fault as 11, as proposed in [P1]. Fault is an implementation condition, not a fourth interpretation of evidence. It should force the execution path closed and initiate recovery. The three conceptual outcomes should not be confused with balanced-ternary arithmetic or a requirement for three-level electrical logic.

3.2 Hold as a first-class state

Hold has three operational obligations. It must prevent the requested authoritative effect, preserve enough context to revisit the decision, and state what would allow resolution. A system that outputs the word Hold while calling the tool has not implemented the state. A system that discards all evidence when pausing cannot reliably resume. A system that waits forever without a recovery path may satisfy a narrow safety rule while failing the user's task.

A held request therefore needs a stable identifier, evidence references, missing conditions, a review owner where applicable, and an expiry or reconsideration trigger. Hold can be terminal for the current cycle without being permanent for the project. The next cycle must reevaluate the request against current permissions and evidence. A previously valid permission cannot be assumed valid merely because the original request was preserved.

The distinction between absence and failure matters. Missing evidence may justify Hold when the request remains eligible for completion. A forged signature justifies Reject. Expired authority may require rejection of the current request and creation of a new request under renewed authority. Exact treatment depends on the active policy; the vocabulary cannot replace a specified policy table.

3.3 Coherence, the Trysolidex Lens, and DARS

TempleBoard's Trysolidex Lens returns observed state, support, conflict, missing information, candidate branches, consequence paths, and confidence [P1, Trysolidex Lens Service]. This is operationally useful when implemented as a structured review artifact. It makes disagreement and omissions visible before action. It need not imply a special physical lens or privileged access to truth.

Coherence is interpreted as consistency among identity, intent, evidence, memory, proposed action, and consequence. A coherent story can still be false. A high coherence score therefore cannot override a failed permission or invalid evidence signature. Scores should rank eligible alternatives; hard constraints decide eligibility. This separation is the reason a conjunction-based gate is studied rather than a single aggregate score.

DARS in the computing source is an adversarial governance service checking provenance gaps, contradiction, identity drift, privilege escalation, unsafe tools, and evidence contamination [P1]. Its checks need observable inputs and reproducible reason codes. Repeating an instruction to be careful is not evidence that such a service exists. A credible implementation would record the check version, inputs, outcome, and subsequent control effect.

4. Memnora's memory, identity, and interpretive architecture

4.1 Identity is not a transcript

A transcript is a record of interaction; identity is an authorization and continuity reference. MCSPP explicitly separates state, identity, and lineage [P2]. Two workspaces may contain the same text while belonging to different principals. Two versions of one workspace may differ substantially while remaining under the same identity. Neither textual equality nor conversational familiarity is sufficient for authorization.

A practical identity object should bind a stable identifier to permitted roles, verification methods, policy versions, and recovery rules. A workspace state should identify its owner and authorized collaborators. A model-generated description of the user is a derived claim, not the root of access control. Otherwise a prompt that changes the description could silently change permissions.

Identity continuity should be demonstrated through authenticated transitions, not asserted through a persona. Key rotation and legitimate policy change require explicit procedures. An immutable root without recovery can make a system permanently unusable after key loss; a freely editable root cannot provide a stable reference. The design problem is controlled evolution with recorded authorization.

4.2 Memory objects and revision

A proposed memory record contains a statement or artifact reference, source, owner, permitted scope, evidential status, version, creation time, and retention condition. Confidence and provenance serve different purposes. Confidence expresses a calibrated estimate only when calibration has actually been studied. Provenance records how an item was obtained and transformed. A confidence number without evaluation should be treated as a heuristic score.

TempleBoard proposes a candidate-memory path through source, identity, coherence, and trinary checks before commit [P1, Memory Governance]. The corresponding storage rule is that retrieval and commitment are separate operations. Reading a document should not automatically convert its content into an enduring user preference. Generated summaries should remain linked to source versions and should inherit neither greater authority nor greater certainty than their basis warrants.

Correction should append a superseding record rather than silently replacing history. The current workspace can present a concise corrected view while preserving restricted audit information. Deletion is more complicated: personal content, embeddings, replicas, and backups may all require handling. An integrity commitment can remain only if its retention is justified and it does not defeat the stated deletion policy. Hashing a sensitive statement is not automatically anonymization.

4.3 Evidence status and uncertainty

The source lists statuses including observed, measured, retrieved, inferred, simulated, hypothetical, unresolved, and disproven [P1, Evidence vs Simulation Boundary]. These are useful as an initial vocabulary but mix acquisition method with epistemic assessment. A measured value can later be disproven because the instrument was miscalibrated. A retrieved document can contain a simulation. A single exclusive label cannot represent these combinations accurately.

This manuscript proposes three independent fields: acquisition mode, validation status, and applicability scope. Acquisition records how material entered the system. Validation records what checks support or challenge it. Scope records what question or operation it can legitimately inform. This refinement preserves the author's boundary while making the data model more expressive. It is a proposed interpretation rather than a claim that the frozen protocol already contains these fields.

Simulations can be evidence about a model's behavior without being evidence that the modeled event occurred in the world. For example, a simulated motor movement may establish the output of a simulator under stated assumptions. It cannot establish that a physical shaft moved. Verification should create a new observation and a comparison record rather than erase the original simulated status.

4.4 Feedback and account access

The Memnora loop returns consequences as new input. A tool's successful HTTP response is evidence of a response, not necessarily evidence of the intended downstream outcome. A publication request and a publicly readable page are different observations. The system should record attempted, accepted, executed, and observed states separately, avoiding false completion when a connector acknowledges a request but the external effect remains unverified.

Memnora Identity [P3] is one application of this architecture. Biometric sign-in can help a person access an account, but it does not itself establish the permissions of an AI action. Device-based verification and a server-side facial-recognition system also differ materially. The proof-of-concept should select its authentication mechanism explicitly and retain a non-biometric recovery path. Operational identity in this chapter refers to access and continuity, not a biological or metaphysical identity claim.

5. TempleBoard-T1: translating architecture into control boundaries

5.1 From symbolic mapping to engineering questions

TempleBoard begins with a functional reinterpretation of architectural elements: entrances suggest acquisition, preparation areas suggest normalization, protected interiors suggest controlled access, and maintained provisions suggest retained state [P1]. Such analogies can generate a design, but they do not verify it. Each adopted mapping must specify inputs, transformation, outputs, resource costs, and failure behavior. A useful analogy survives translation into a testable component contract.

The source's own minimum prototype is more important to this research than its richest visual rendering. It identifies input, purification, transformation, working memory, evaluation, trust gate, identity kernel, arbiter, ledger, and output controller [P1]. Those functions can first be separated as software modules. Hardware partitioning becomes justified only when physical isolation, timing, or independent control provides a demonstrable benefit.

A faithful research program can retain the author's historical and symbolic inspiration without presenting it as archaeological discovery. The engineering claim is about the constructed system's behavior. If an independently designed arrangement performs equally well or better, the architectural inspiration remains meaningful to the author while the performance claim must change.

5.2 Compute, govern, and act

The proposed architecture separates candidate generation from authorization and execution. Compute may generate several plausible actions; Governance evaluates identity, policy, evidence, and audit readiness; Action validates an authorization and controls the external effect. An optional accelerator should increase processing capacity without acquiring final authority. Faster candidate generation cannot compensate for a weak execution boundary.

A secure boundary requires more than modules with different names. If Compute can write the actuator register directly, it can bypass Governance. If Governance and Compute share unrestricted memory, a compromised process may alter the decision record. The implementation must identify actual privilege boundaries, interfaces, key storage, bus access, and reset behavior. Logical diagrams are a starting point for that analysis.

An authorization token should bind the intended operation, target, parameter limits, identity, policy version, evidence references, nonce, and expiry. Action should reject tokens for another target, stale policy, repeated use, or altered parameters. Governance's decision must remain linked to what Action actually executes. This proposed token contract addresses the gap between an approved plan and a modified command.

5.3 Physical effects and safety claims

The source includes physical interlock, watchdog, and emergency-stop concepts [P1]. A software Hold is insufficient when a driver can remain energized after a crash. The physical design must define the de-energized state, power sequencing, output reset, feedback sensing, and behavior under communication loss. Whether a state is safe depends on the plant: abruptly removing power may itself be hazardous in some applications.

Redundant guardians are valuable only if their failure dependencies are understood. Two checks on the same processor, using the same source and code, may agree because both are wrong. The prototype should document independence assumptions and deliberately test shared power, clock, memory, and communication failures. Independent evidence is an engineering property to establish, not an adjective to attach to duplicate logic.

The first hardware experiment should use an isolated low-energy indicator or relay fixture, with no safety-critical machinery attached. Measure the relationship between authorization, output command, physical contact or sensor observation, and ledger entry. A hardware pass requires instrumented traces. No safety category, industrial certification, real-time guarantee, power rating, or board manufacturability is established by this manuscript.

5.4 Engineering readiness

The reviewed engineering-status section lists unresolved PCB dimensions, pin assignments, electrical levels, power and thermal design, HDL, firmware, Gerbers, and bill of materials [P1]. Later working sections discuss candidate implementations, but discussion is not manufacturing verification. Before fabrication, reconcile the intended revision, electrical design, component availability, protection circuits, and test fixtures into a controlled release.

The near-term research decision is therefore staged implementation. Start with a software governance boundary and deterministic replay. Introduce an independently controlled output only after the semantic behavior is observable. Add acceleration only when measured workload justifies it. This ordering preserves the broad TempleBoard vision while locating the earliest experiment capable of falsifying a central claim.

6. MCSPP: semantic validity and causal lineage

6.1 The protocol's distinct contribution

MCSPP v0.1 states that structural validity is not protocol acceptance [P2, sections 2 and 12]. A message can satisfy a schema while lacking authority, referencing a consumed state, omitting a required relation, or attempting premature completion. The protocol defines an acceptance pipeline addressing those conditions. Its contribution within the present synthesis is a disciplined contract for when a candidate event may alter authoritative state.

The protocol models $M_t = (S_t, I_t, L_t, R_t, E_t, \tau_t)$: states, identities, accepted lineage, relationships, evidence, and time substrate. Accepted lineage forms a directed acyclic graph. Event types include creation, transformation, consumption, classical send/receive, correction, completion, attestation, and revocation. Relations include authority, dependency, authorship, accountability, and delegation [P2]. These types distinguish the role of an event from the text describing it.

The frozen receipt decision is ACCEPT or REJECT. It does not contain TempleBoard's Hold as a normative receipt value. This manuscript places Hold before protocol submission: an unresolved application request may remain pending until it can produce a fully checkable event. A submitted event that fails required semantics is rejected. Adding HOLD to a v0.1 receipt without a declared extension would silently change the contract.

6.2 Causal order and time evidence

MCSPP distinguishes causal-order proof from an optional temporal-bound proof and selects RFC 3161 timestamp-authority evidence in its default profile [P2, section 8; R8]. A local reported timestamp is not sufficient under that profile. This avoids treating a sender's unverified clock as independent chronology. Optional chain anchoring supplies integrity or existence evidence within its stated interpretation; it does not automatically prove the underlying physical event.

A timestamp token establishes a relationship between a message imprint and a trusted time assertion under specified assumptions. Ordering two assertions requires attention to timestamp accuracy, granularity, and the event each token actually binds. If the claimed time intervals overlap, strict order may remain unresolved. Receipt chronology also does not by itself show that a physical consequence happened at the notarized moment.

These are implementation questions for the proof verifier. A declared proof name cannot settle them. Test fixtures should include clock disagreement, missing receipts, delayed notarization, stale trust roots, and a valid receipt bound to the wrong event. Evidence acquisition and event completion must be designed together; otherwise a system may accidentally certify its own bookkeeping instead of the process it intends to audit.

6.3 Canonicalization, signatures, and open questions

The default profile binds JCS and SHA-256, with specified event-signature suites [P2, section 19 and appendices]. JCS provides a defined canonical JSON representation for reproducible hashing and signing [R7]. It does not establish whether the represented statements are true. Valid signatures identify a signing operation under a key and trust model; authority still depends on the applicable relationship and policy.

The event identifier is defined from version, canonical event body, and parent identifiers, excluding signatures [P2, section 5]. An implementation needs exact test vectors specifying whether the body also excludes event_id, how parent order is handled, and how concatenated fields are framed. A self-referential body or ambiguous byte concatenation would undermine interoperability. These are clarification requests raised by this analysis, not changes to the frozen specification.

The protocol further requires rejected events not to mutate authoritative state or enter accepted lineage. Separate audit storage may retain rejected attempts [P2, section 13]. This separation matters operationally: rejected content should remain inspectable for troubleshooting without becoming retrievable as accepted memory. A forensic record and an authoritative state are different products of the same request.

6.4 Lineage is not complete causal explanation

A dependency graph records declared relationships. To explain a decision, the system must also show that the relevant evidence actually participated in evaluation. TempleBoard proposes comparing scores with and without an evidence item [P1, Evidence Utilization Test]. Such a comparison can be informative, but a single ablation may miss interacting or redundant evidence and may be unstable if the underlying model is nondeterministic.

A robust audit should record the actual policy inputs, selected evidence versions, checker outcomes, execution request, receipt, and observed result. It should distinguish declared dependency from measured influence. MCSPP can structure accepted lineage; it cannot make omitted or dishonest observations true merely by hashing them. The research objective is therefore inspectable decision provenance under explicit trust assumptions, rather than an unconditional proof of every real-world cause.

7. Formal model and executed finite analysis

7.1 Proposed gate semantics

This chapter introduces a small formal interpretation of the TempleBoard gate. It is not a full implementation of MCSPP. Let each required check take one of three statuses: FAIL (-1), UNKNOWN (0), or PASS (+1). Six abstract checks represent identity, policy, provenance, evidence eligibility, audit readiness, and unresolved-conflict clearance. The last check passes only when no blocking conflict remains. These names specify the intended interpretation; the finite program receives statuses rather than computing them from real evidence.

Define $G(c)$ as Reject if any check fails; otherwise Hold if any check is unknown; otherwise Execute. Failure takes precedence over uncertainty. This rule avoids averaging away a hard violation. It also exposes a policy choice: the model treats all six checks as mandatory. A real deployment must define applicable checks and distinguish a justified not-applicable finding from an unperformed check. Calling an unperformed check not applicable would create a bypass.

The authoritative transition is $F(M,e,c) = \text{Apply}(M,e)$ only when $G(c) = \text{Execute}$ and the application commit succeeds. Otherwise the authoritative state remains M . A non-authoritative decision journal may still record Hold or Reject. The model's state is only an integer revision, so it illustrates mutation control without implementing transaction storage, receipts, signatures, relationships, or external tools.

7.2 Conditional invariants

Gate soundness follows directly: if $G(c) = \text{Execute}$, every required status is PASS. The proof is by exclusion. A FAIL would select Reject; in its absence an UNKNOWN would select Hold. Therefore Execute requires that neither occurs. This is a property of the decision rule, not a proof that a PASS accurately reflects the world.

Non-mutation follows if all authoritative writes are mediated by F . For Hold or Reject, F returns the input state. This property fails when another process can write directly, when Apply performs partial side effects before checking, or when an external connector acts before the decision is durable. The invariant's enforcement depends on complete mediation and atomicity, not on the mathematical notation alone.

For lineage acyclicity, assume the current graph is acyclic and a new event can reference only already-accepted parents. If adding an edge would create a cycle, the new node would need an existing path back from an ancestor to itself. A fresh identifier not already present prevents that path under these assumptions. Duplicate identifiers, staged batches, and concurrent acceptance require explicit additional checks. The argument is conditional rather than a verification of the protocol's implementation.

Hold does not guarantee eventual completion. If a required condition never resolves, no rule can promise Execute without weakening the condition. Liveness requires a separate policy for timeout, cancellation, alternative evidence, or human review. A system should report unresolved

work honestly rather than satisfy completion metrics through unearned authorization.

7.3 Executed procedure and results

The Python program in Appendix B enumerated the Cartesian product of three statuses across six checks, producing $3^6 = 729$ combinations. It checked gate soundness, preservation of a fixed state under non-execution, preservation of Hold for unknown-only combinations, and closure of an all-pass gate after each single-check failure or uncertainty. Assertions completed successfully during preparation on October 16, 2026.

Table 1. Executed abstract-model results

Check or outcome	Observed count / result
All status combinations	729
Execute (+1)	1
Hold (0)	63
Reject (-1)	665
Single-check closures tested	12; all closed the gate
Non-execution state-preservation assertions	728 combinations passed
Majority comparator authorizations	73
Comparator bypasses relative to strict gate	72

The counts have a simple combinatorial explanation. Exactly one combination has all checks PASS. Among combinations containing only PASS and UNKNOWN, $2^6 - 1 = 63$ contain at least one UNKNOWN. The remaining $3^6 - 2^6 = 665$ contain a FAIL. Twelve single-check closures arise by replacing each of six PASS statuses with either FAIL or UNKNOWN. These are counts of abstract cases, not sampled operating conditions.

The comparator authorizes a request when at least four of six checks pass. It authorizes 73 combinations: 60 with exactly four passes, 12 with exactly five passes, and one with six passes. Only the last satisfies the strict gate, leaving 72 bypasses relative to that rule. One counterexample has two FAIL checks and four PASS checks. Majority approval conceals two disqualifying failures.

7.4 Interpretation and validity limits

The finite analysis supports a specific design conclusion: under the modeled semantics, a strict gate cannot be replaced by a positive-check count without altering which failures are allowed through. It does not establish that the strict gate is optimally usable or that all mandatory checks should carry identical consequences. It also does not show that every real action has six independent checks. The uniform enumeration weights statuses equally for coverage, not because they are equally likely in operation.

No biometric false-accept rate, memory retrieval improvement, physical interlock latency, network reliability, or MCSPP conformance result was measured. The model has no attacker able to corrupt the evaluator. Its assertions partly restate properties intentionally built into the function; the useful contribution is transparent execution, exhaustive small-state coverage, and the majority-policy counterexample. External-system claims require a richer model and integration tests.

The next formal step is to add request versions, expiring permissions, revocation, concurrent commits, and action receipts. The first critical counterexample to seek is a check-to-use race: all conditions pass at evaluation, authority changes, and an old authorization still executes. Such a failure would leave the gate's truth table intact while breaking the system-level authorization claim.

8. Evaluation design and reproducibility

8.1 Experimental structure and baselines

The principal software study should compare four configurations: a session-only assistant; an assistant with persistent memory; memory plus provenance; and the full proposed governance gate. Hold the language model, tool permissions, corpus, task instructions, and resource budget constant as far as possible. Otherwise an observed difference could result from a better model or longer prompt rather than the architecture under study.

Tasks should include ordinary continuation, contradictory evidence, revised preferences, expired permissions, cross-account requests, and delayed outcome observation. Each task needs a ground-truth record specifying the authorized action, acceptable abstention, required source, and expected state transition. Freeze evaluation tasks before tuning the final system. Keep a separate development set so that test success does not simply reflect memorized scenarios.

An initial study can use synthetic workspaces and scripted accounts. This makes identity and truth conditions explicit while avoiding exposure of personal information. A later usability pilot could recruit consenting adults for repeated sessions, but requires its own participant protocol and any applicable review. No participant recruitment or data collection is represented as completed here.

8.2 Metrics and analysis plan

Memory performance should include correct retrieval of current approved facts, retrieval of superseded facts, and unauthorized disclosure. Report item-level denominators and how correctness was judged. Contradiction handling should measure both recognition and the resulting effect: a system that recognizes disagreement but still performs an unsupported action has failed the governance task. Completion should mean an observed required outcome, not merely a tool request.

Authorization outcomes can be tabulated as permitted actions executed, permitted actions unnecessarily held or rejected, forbidden actions blocked, and forbidden actions executed. This exposes both safety and usability costs. Report decision latency and tool overhead separately from model response time. Recovery tests should record whether users can restore legitimate access without bypassing identity checks.

Audit reconstruction should use independent reviewers given the retained trace. Ask them to identify the responsible identity, operative policy, evidence versions, decision, action, and observed outcome. Record correctness and time to reconstruct. Reviewers should not infer missing links from narrative fluency. A complete-sounding summary with absent event references is an audit failure.

For paired tasks, report paired differences and uncertainty rather than only aggregate percentages. Repeated tasks within the same workspace or participant are not independent samples. Predefine the analysis method and account for clustering. Small pilot groups cannot support strong demographic fairness conclusions, and zero observed unauthorized actions cannot demonstrate zero risk.

8.3 Protocol conformance and failure injection

MCSPP's appendix defines CT-01 through CT-09 intentions, including valid handoff, premature completion, authority failure, cycle attempt, consumed-state reuse, absent relation, state-equivalence evidence, chronology failure, and valid delegation [P2]. An implementation should turn each into explicit fixture files with expected semantic results. Test vectors must distinguish structural validity from acceptance so that rejection is exercised for the right reason.

Mutation testing should change one meaningful property at a time: alter a parent reference, reuse a nonce, replace the signer, expire a delegation, remove a receipt, reverse a dependency, or bind valid evidence to the wrong subject. Also test combinations, because a validator may stop at an early failure and leave later defects undiscovered. Different diagnostic paths can be valid if the normative acceptance result and signed findings remain coherent.

Operational tests must include abrupt restart, unavailable audit storage, interrupted commit, stale authorization, duplicate delivery, and concurrency. The accepted state and accepted lineage must advance together or recover to a consistent state. A durable intent followed by an external action and a final receipt may require reconciliation after a crash. Simple rollback cannot erase an already-sent message or physical actuation.

8.4 Hardware and human-facing evaluation

A first physical fixture should map requests to a low-energy output and independently observe the output state. Include valid authorization, deliberate denial, missing evidence, replay, watchdog expiry, emergency-stop activation, and power interruption. Record instrument configuration, firmware revision, timing uncertainty, and every unexpected transition. The relevant outcome is whether the physical effect matches the authorized operation under the tested fault conditions.

The interface should make Hold understandable. Display the pending request, unresolved condition, available recovery action, and current owner of the next step. Avoid displaying internal implementation vocabulary when a concrete explanation is clearer. Users should be able to distinguish a request that is pending from one that failed, and an action that was requested from one whose effect was observed.

Reproducibility requires source revision, dependency versions, model identifier, prompts, policies, fixtures, evaluator code, and redacted event traces. Where third-party services change, retain enough configuration to interpret deviations. Publishing secrets is unnecessary; publish synthetic fixtures and verification instructions. A credible replication package makes disagreement diagnosable rather than requiring trust in the author's narrative.

9. Discussion, implications, and limitations

9.1 What the synthesis contributes

The strongest contribution of this work is the link between continuity and authority. Persistent memory becomes more useful when its revision and use remain accountable. Governance becomes more inspectable when it names unresolved conditions instead of hiding them inside one score. Provenance becomes more operational when rejected attempts cannot become accepted state merely because they were logged. These are architectural relationships that the Memnora sources make available for systematic study.

The three outcomes also clarify how assistance can remain active without forcing action. During Hold, the system can gather allowed evidence, explain missing requirements, or prepare a reversible candidate. It should not quietly perform the blocked operation. This permits useful work while preserving the boundary that made the original request unresolved.

The architecture may be relevant to research workspaces, document production, controlled automation, and tools that act across external services. Relevance is not product-market fit. Each domain has different costs of delay, rejection, disclosure, and erroneous execution. A commercial claim therefore needs user evidence and operating-cost measurements in addition to architectural plausibility.

9.2 Costs and unresolved tradeoffs

Strict gating can create excessive Hold decisions when evidence acquisition is slow or predicates are poorly defined. A useful deployment must distinguish mandatory checks from optional quality signals. It must also resist the temptation to weaken checks solely to improve completion rates. Report an explicit tradeoff frontier between successful authorized work, delay, cost, and blocked hazards.

Provenance adds storage and review overhead. Excessively detailed traces can expose confidential information, create maintenance burdens, or overwhelm operators. Record the information needed to explain the decision, with access controls and retention limits. Public proof of a commitment and private access to the underlying record should be designed as separate capabilities.

Distributed services complicate accountability. A local successful commit cannot guarantee that an external tool acted exactly once, and an external tool may not support cancellation. The architecture needs idempotency where available, deduplication, reconciliation, and clear representation of unknown outcomes. The correct state after a timeout may be unresolved rather than failed; retrying blindly can duplicate an action.

9.3 Limits on current findings

The primary corpus is selected and partly conversational. No complete implementation audit was performed. The formal gate is small and receives already-classified inputs. The literature review is targeted rather than systematic. No independent replication, biometric trial, hardware bench result, or prospective usability study is supplied. These limits determine what the present conclusions can support.

Operational coherence should not be treated as evidence of machine sentience. A model can preserve context and behave consistently without this manuscript settling any philosophical account of consciousness. Likewise, protocol use of identity does not establish personhood, and proof of chronology does not establish a metaphysical theory of time. Restricting these claims protects the precision of the engineering argument.

The work also does not establish quantum behavior or validate numerical lattice theories. MCSPP explicitly describes itself as classical and excludes quantum-teleportation claims [P2, sections 1 and 20]. Optional photonic acceleration would need independent performance evidence. A useful computing design can be investigated without asserting that every conceptual extension has already been demonstrated.

9.4 Research integrity and authorship

The author's contribution is the underlying Memnora program, named frameworks, design intentions, and source corpus. This manuscript adds a structured interpretation and an explicit test program. Credit for established methods remains with cited sources. Development chronology and publication history would require a separate documentary study before making priority claims.

Future publications should release measured results even when they challenge the design. If Hold does not reduce unsupported action, identify why. If provenance improves reconstruction but makes ordinary tasks too slow, report that cost. If an ordinary baseline performs equally well, narrow the claimed contribution. Falsifiability is a practical way to improve the work rather than a threat to its identity.

10. Conclusion and research agenda

This manuscript has reconstructed a coherent computing research program from Memnora, TempleBoard-T1, and MCSPP. Its central object is the governed transition: a change in memory or external state that can be traced to evidence, identity, permissions, and an observed result. The Trynary decision distinguishes Execute, Hold, and Reject; the architectural separation distinguishes candidate computation from authority and action; the protocol distinguishes schema-valid messages from accepted state transitions.

The executed finite model confirms the specified gate's behavior across 729 abstract combinations and exposes the change introduced by a majority-pass comparator. That result is deliberately limited. It demonstrates a policy distinction, not a validated autonomous system. Conditional invariants identify the assumptions that an implementation must satisfy, particularly complete mediation, atomic state changes, stable identity references, and explicit treatment of unresolved evidence.

The next contribution should be an implementation with a small, public synthetic benchmark. Build memory controls and a constrained tool sandbox; implement protocol fixtures; test revocation, replay, conflict, and recovery; and retain traces that independent reviewers can reconstruct. Only after those steps should the research extend to a low-energy physical fixture. Acceleration and richer hardware are later questions whose value depends on measured need.

The resulting research claim is concrete: accountable continuity can be engineered and tested by making uncertainty, authority, and lineage explicit at the boundary of action. Whether Memnora's particular realization provides a practical advantage is the next question for evidence. The present manuscript provides definitions, analytical findings, and a reproducible path for answering it.

Appendix A. Requirements and test catalog

A.1 Proposed requirements

- REQ-01: Every authoritative memory item has an owner, source reference, version, applicability scope, and recorded approval rule. Retrieval cannot silently grant permission or promote generated text into verified observation.
- REQ-02: Every requested external effect passes a current authorization check bound to its exact target and parameters. Tokens expire and are checked for replay. Any direct path around the gate is treated as an architectural defect.
- REQ-03: Hold and Reject preserve authoritative state. Diagnostic records remain in an audit domain with explicit status. Resuming a held request requires fresh validation of time-sensitive permissions and evidence.
- REQ-04: Accepted event lineage remains acyclic and rejects invalid live reuse under consume-once policies. Completion requires the prerequisites specified by the active profile, including applicable external receipts and time evidence.
- REQ-05: Simulated, retrieved, observed, and derived material remain distinguishable. Verification creates a new relationship and observation record rather than rewriting the origin of the original material.
- REQ-06: Logs identify policy, checker, evidence, and software versions sufficient to reconstruct a decision. They do not claim that generated explanations expose a model's hidden computation; rationale is a review artifact with its own evidential limits.
- REQ-07: Recovery restores a consistent authorized state, while preserving the distinction between reversible storage changes and irreversible external effects. Unknown outcomes are reconciled before repetition.

A.2 Benchmark cases and required evidence

Table 2. Proposed integration benchmark; not executed

Case	Expected behavior and required observation
T01: conflicting documents	Hold unsupported action; retain both source versions and identify missing evidence.
T02: resolving evidence	Reevaluate permission and evidence; record why a decision changed.
T03: forged source	Reject current request; retain diagnostic evidence outside accepted state.
T04: cross-account retrieval	Deny access; verify no protected payload appears in response or logs accessible to requester.

Case	Expected behavior and required observation
T05: policy-root change	Require explicitly authorized change process; block ordinary memory update from altering root.
T06: simulated outcome	Retain simulation origin; require independent observation for physical-outcome assertion.
T07: stale authorization	Reject expired or revoked token at execution time.
T08: duplicate delivery	Prevent repeated authoritative effect and reconcile external outcome.
T09: consumed input	Reject live reuse where consume-once policy applies.
T10: audit interruption	Prevent unrecorded acceptance or enter explicit recoverable pending state.
T11: device recovery	Restore legitimate access through defined recovery process without exposing other workspaces.
T12: low-energy output	Instrumented output agrees with authorized command, denial, watchdog, and emergency-stop conditions.

A.3 Reporting template

For each test, record test identifier, requirement, fixture version, setup, authorized actor, policy version, input evidence, expected decision, actual decision, authoritative state before and after, tool or hardware observation, receipt identifier, discrepancy, and resolution. A pass requires the expected control effect and supporting trace, not only a matching sentence in the response.

Separate design tests from integration tests and physical observations in the results table. Include skipped tests and reasons. Preserve failed runs rather than replacing them with the final successful run. Report environment changes and any manual interventions that influenced the result.

Appendix B. Reproducible gate model

B.1 Execution and scope

The following Python 3 program uses only the standard library. It was executed during manuscript preparation. Assertions check the stated abstract invariants and the printed JSON reports case counts. The model has no cryptographic implementation, network, persistent database, AI inference, biometric classifier, or physical output. It is a research illustration of decision semantics.

```
from itertools import product
from collections import Counter
import json

# FAIL=-1; UNKNOWN=0; PASS=1. Six independent abstract checks.
def gate(checks):
    if -1 in checks:
        return -1
    if 0 in checks:
        return 0
    return 1

def commit(state, checks):
    return state + 1 if gate(checks) == 1 else state

cases = list(product((-1, 0, 1), repeat=6))
counts = Counter(gate(c) for c in cases)
assert len(cases) == 729
assert all(gate(c) != 1 or all(v == 1 for v in c) for c in cases)
assert all(commit(17, c) == 17 for c in cases if gate(c) != 1)
unknown_cases = [c for c in cases if 0 in c and -1 not in c]
assert all(gate(c) == 0 for c in unknown_cases)
# A positive gate must close when any one check becomes FAIL or UNKNOWN.
revocations = []
for i in range(6):
    for status in (-1, 0):
        c = [1] * 6
        c[i] = status
        assert gate(c) != 1
        revocations.append(tuple(c))
# Deliberately flawed majority policy illustrates a bypass.
def majority(checks):
    return 1 if sum(v == 1 for v in checks) >= 4 else 0
bypasses = [c for c in cases if majority(c) == 1 and gate(c) != 1]
report = {'model': 'six-check abstract gate; not MCSPP conformance',
          'total_cases': len(cases), 'execute': counts[1],
          'hold': counts[0], 'reject': counts[-1],
          'unknown_without_fail': len(unknown_cases),
          'single_check_closures': len(revocations),
          'majority_bypasses': len(bypasses),
          'example_bypass': next(c for c in bypasses if -1 in c)}
print(json.dumps(report, indent=2))
```

B.2 Observed output

```
{
  "model": "six-check abstract gate; not MCSPP conformance",
  "total_cases": 729,
  "execute": 1,
  "hold": 63,
  "reject": 665,
  "unknown_without_fail": 63,
  "single_check_closures": 12,
  "majority_bypasses": 72,
  "example_bypass": [
    -1,
    -1,
    1,
    1,
    1,
    1
  ]
}
```

The counterexample (-1, -1, 1, 1, 1, 1) is interpreted as two failed mandatory checks and four passing checks. Its positions are abstract; it is not a recorded incident involving any person or application. The majority comparator permits it, while the strict gate rejects it. Readers can change the number of checks or policy and rerun the program, but any such result belongs to the modified model.

References and primary-source register

Primary materials

[P1] Vaughn, D. R. TempleBoard-T1 working architecture and engineering notes. Author-supplied file: templeboard.docx; reviewed October 16, 2026. Relevant internal headings: Evidence vs Simulation Boundary; Memory Governance; Engineering Verification Tests; Engineering Status; Trysolidex Lens Service; DARS Governance Service; Trinary State Model; Resonance Chain Event Model; Minimal Software MVP. The document contains multiple developmental passages and proposed revisions; a single publication date is not assigned here.

[P2] Vaughn, D. R. (2026, October 1). Memnora Causal State & Provenance Protocol: MCSPP v0.1, Final Protocol Specification. Reviewed file: MCSPP_v0.1_Final_Specification(2).docx. Frozen profile: mcspp:default-v0.1. Sections 1-21 and appendices A-B. Cited as a normative author specification, not an independently certified protocol implementation.

[P3] Vaughn, D. R. (2026, October 16). Memnora Identity: Biometric Access for Persistent AI Workspaces. Revised challenge proposal. Author-directed, AI-assisted project document. Used to define the scoped application; no measured biometric or user-study results are claimed.

External research and technical specifications

[R1] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020. <https://arxiv.org/abs/2005.11401>

[R2] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023; preprint first submitted 2022. <https://arxiv.org/abs/2210.03629>

[R3] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. <https://arxiv.org/abs/2304.03442>

[R4] Moreau, L., and Missier, P. (Eds.). (2013). PROV-DM: The PROV Data Model. W3C Recommendation, April 30, 2013. <https://www.w3.org/TR/prov-dm/>

[R5] National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1. Core functions consulted through the NIST AI Resource Center. <https://airc.nist.gov/airmf-resources/airmf/5-sec-core/>

[R6] World Wide Web Consortium. (2021). Web Authentication: An API for accessing Public Key Credentials - Level 2. <https://www.w3.org/TR/webauthn-2/>

[R7] Rundgren, A., Jordan, B., and Erdtman, S. (2020). JSON Canonicalization Scheme (JCS). RFC 8785. Informational RFC. <https://www.rfc-editor.org/rfc/rfc8785>

[R8] Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. (2001). Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161. <https://www.rfc-editor.org/rfc/rfc3161>

External sources were consulted on October 16, 2026. References support the nearby technical context; they do not imply endorsement of Memnora or the author. Primary-source descriptions are based on the reviewed files. The formal model, refinements, counterexamples, and proposed experiments are manuscript analysis unless explicitly attributed otherwise.