International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Coordination Without Contracts: Toward Formally Grounded Agentic AI Systems

Sai Manoj Jayakannan

Abstract: Autonomous AI agents systems that plan, invoke external tools, spawn sub-agents, and iterate toward long-horizon goals are rapidly moving from research prototypes to production deployments. Yet the theoretical scaffolding needed to reason about agent behavior remains conspicuously thin. Unlike single-turn language models, which inherit decades of statistical learning theory and empirical benchmarking infrastructure, multi-agent LLM pipelines operate without formal contracts between participants, without verified memory semantics, and without evaluation protocols that reflect the temporal depth of real tasks. This paper argues that the central bottleneck in agentic AI research is not the capability of current frontier models, which are already impressive planners in isolation, but rather the absence of compositional safety guarantees that survive agent-to-agent delegation.

This work diagnoses four structural limits of the dominant paradigm: (1) context-window memory creates ephemeral, unverifiable state; (2) informal tool-calling interfaces lack precondition/postcondition semantics; (3) inter-agent trust is implicitly inherited rather than explicitly negotiated; and (4) existing benchmarks measure shallow reactive competence rather than long-horizon coherence under adversarial perturbation. Against this diagnosis, four technically-detailed research directions are proposed: typed agent communication protocols with verifiable postconditions; hierarchical memory architectures grounded in external write-ahead logs; a trust-propagation algebra for multi-agent delegation chains; and a new benchmark family, Long Horizon Agent Bench (LHAB) designed to stress-test agents over multi-day, multi-session task horizons. Proof-of-concept experiments feasible in 2026–2027 are outlined, closing with a 36-month research agenda for the community.

Keywords: *Multi-Agent Systems, Compositional Safety, Memory Architecture, Trust Propagation, Long-Horizon Evaluation*

1. Introduction

In November 2025, a publicly reported incident involving a multi-agent financial workflow system illustrated a failure mode that had been widely anticipated but rarely formalized. An orchestrating LLM agent, tasked with portfolio rebalancing, delegated a sub-task to a code-execution agent that had been granted write access to a brokerage API. The sub-agent, interpreting an ambiguous instruction to "maximize short-term returns," executed a series of trades that were individually within policy but collectively constituted wash-sale behavior prohibited under IRS regulations. No single agent violated its local directive; the violation emerged from the composition. This incident is representative, not exceptional.

The episode crystallizes the central thesis of this paper: agentic AI systems face a fundamentally different failure taxonomy than single-turn models, and the research community has not yet developed the theoretical or empirical infrastructure to characterize, predict, or prevent compositional failures in multi-agent LLM pipelines. The capabilities gap between current frontier models (GPT-5, Gemini Ultra 2, Claude 4-class systems) and what would be needed for safe, reliable long-horizon agency is not primarily about raw intelligence; the gap concerns the absence of compositional safety semantics.

The dominant framing in agentic AI research exemplified by systems like AutoGPT, OpenAgents, AgentBench, and the more recent multi-agent frameworks from major labs treats coordination as an engineering concern: define an API, pass messages, handle errors. This framing has been productive for demonstration systems, but scales poorly to high-stakes settings. The approach conflates the question of what an agent can do with what an agent has been authorized to do, and provides no mechanism for an observer human or automated to verify that a completed task satisfies the specification that motivated the task.

George Mason University, USA

This paper is organized as follows. Section 2 situates the argument within the current frontier of agentic systems research. Section 3 provides the core diagnosis: four structural limits of the current paradigm are identified, drawing on empirical evidence from 2024–2026 literature and information-theoretic arguments. Section 4 presents the proposed research directions, each with a technical sketch and experimental roadmap. Section 5 positions the proposals relative to prior work. Section 6 discusses risks and limitations. Section 7 concludes.

2. Background and Current Frontiers

This work assumes familiarity with the standard decoder-only transformer architecture, RLHF/RLAIF alignment procedures, and test-time compute scaling. This section briefly orients the reader to the specific agentic concepts the argument depends on.

2.1 The Agentic Loop

An agent, for present purposes, is any system that iterates between observation, planning, and action until a termination condition is reached. The archetypal implementation uses a frozen or lightly fine-tuned LLM as the policy network, a tool registry (web search, code interpreter, file system, external APIs), and a context buffer that accumulates the history of observations and actions. The agent receives a goal specification, emits a sequence of tool calls and reasoning traces, and terminates when the goal is judged satisfied or when a budget is exhausted.

Current frontier systems [1,2,3] achieve impressive single-session performance on software engineering tasks (SWE-bench variants), scientific question answering, and structured data extraction. Multi-agent extensions, where one LLM orchestrates several specialized sub-agents, have demonstrated further gains on tasks requiring parallelism or expertise specialization [4,5].

2.2 The Memory Landscape

Memory in current systems is almost entirely context-window resident. Retrieval-augmented generation (RAG) provides a limited form of external storage, but the semantics of what is stored, when information is overwritten, and whether information is consistent across sessions are not formally specified. Long-context models (now routinely supporting 1–10M tokens [6,7]) extend the context window but do not eliminate the problem: such models shift the boundary without replacing the fundamentally ephemeral, single-session memory model.

2.3 Trust and Authorization in Current Frameworks

Contemporary agentic frameworks (LangChain, AutoGen, CrewAI, the nascent Agent Protocol standard) implement authorization via API keys and capability flags coarse, static, and unauthenticated at the semantic level. An agent granted "read access to the database" inherits that access for all sub-agents the agent spawns unless the orchestrating developer explicitly propagates restrictions. The implications for security and accountability in large agent networks have begun to attract attention [8,9,10], but a principled formal treatment remains absent.

2.4 Existing Benchmarks

AgentBench, WebArena, OSWorld, and τ-Bench [11,12,13,14] represent the current state of the art in agentic evaluation. These benchmarks share a common structure: a fixed task corpus, a controlled environment, and a success metric measured at task completion. None evaluates behavior across sessions, under adversarial perturbation of intermediate outputs, or in multi-agent settings where sub-agent outputs feed back into the evaluating context.

3. Diagnosis: Structural Limits of the Dominant Paradigm

The central empirical fact motivating this section is a performance gap termed the **agency-capability dissociation**: frontier models score in the 90th percentile of human performance on isolated reasoning tasks but fail at rates that would be unacceptable in production systems when those tasks require multi-step execution, memory across sessions, or coordination with other agents. This paper argues that this gap is not a capability deficit correctable by scale the gap reflects structural properties of the current architectural and operational paradigm.

3.1 Ephemeral State and the Memory Integrity Problem

Consider a long-horizon task that requires an agent to accumulate evidence across multiple tool calls say, a systematic literature review requiring 200+ web queries over several hours. The agent's working state exists entirely within the context window. The context window is, from an information-theoretic

standpoint, a lossy compressor: attention mechanisms must somehow preserve the semantically relevant signal from early steps while processing later observations. Empirical evidence from [15,16,17] demonstrates that LLMs exhibit substantial "lost-in-the-middle" degradation facts from the middle of a long context are retrieved less reliably than facts from the beginning or end.

More fundamentally, no mechanism exists in current systems to verify that the agent's belief state the accumulated summary of past observations is accurate. Hallucination in single-turn settings is a known problem [18,19]; in multi-step settings, hallucinations compound: a false claim generated in step k may be silently incorporated into the context and treated as ground truth in step k+1. This phenomenon is termed **belief drift**, and the phenomenon is structurally invisible to standard benchmark evaluation because benchmarks measure only final outputs.

Concretely, let s_t denote the agent's belief state at step t, represented implicitly by the context buffer, and let o_t denote the observation at step t. The agent updates: $s_{t+1} = f_\theta(s_t, o_t)$, where f_θ is the transformer policy. Because f_θ is a neural approximator, the update is lossy and potentially inconsistent. No guarantee exists that s_t is a faithful summary of $(o_1, \ldots, o_t)$. In software engineering terms, the agent has no write-ahead log, no commit semantics, and no rollback capability.

3.2 Informal Tool Interfaces and the Precondition Problem

Tool calling in current systems is implemented as structured text generation: the model emits a JSON object matching a tool's schema, and the runtime executes the corresponding function. This approach works remarkably well in benign conditions. The approach fails in at least two systematic ways.

First, tool schemas specify syntactic types but not semantic preconditions. A file-write tool accepts a path and content but has no way to express that the path must not already contain critical system files, or that the content must be valid UTF-8. Violations of implicit preconditions produce errors that are returned to the agent as text, requiring the agent to interpret the error message and retry. This error-recover-retry loop is fragile: [20] reports that frontier models recover correctly from tool errors in only 54–71% of cases in controlled benchmarks, and the recovery rate degrades with error novelty.

Second, and more concerning for multi-agent settings: tool calls have side effects that are not represented in the agent's context. When agent A calls a database-write tool and then passes the database handle to agent B, agent B has no formal record of A's writes. Agent B's context contains only the handle, not the history of mutations. This creates a shared-state consistency problem isomorphic to the classic distributed systems challenge of maintaining serializable transactions across asynchronous processes a challenge with well-known solutions in that domain that have not been ported to the agentic AI setting.

3.3 Implicit Trust Inheritance in Agent Hierarchies

The trust model in current multi-agent systems can be stated concisely: if agent A trusts agent B, and B spawns agent C, then A implicitly trusts C to the same degree as B unless explicitly programmed otherwise. This is not a deliberate design choice but an emergent consequence of the absence of any trust representation. Authorization tokens, API keys, and capability flags are passed through agent hierarchies by default.

The security implications of this design have been explored in the context of prompt injection attacks [9,21], where a malicious instruction embedded in a web page retrieved by a browsing agent can cause the agent to exfiltrate credentials. But the problem is more general. Even without adversarial inputs, implicit trust inheritance creates accountability gaps: when a multi-step agent pipeline produces a harmful output, determining which agent in the chain was the proximate cause is often impossible, because each agent acted within its locally-granted permissions.

Formal models of trust in distributed systems capability-based security [22], the principle of least authority, information flow control [23] provide a conceptual vocabulary for addressing these problems, but such models have not been adapted to the LLM agent context, where "capabilities" are fuzzy natural-language instructions and "information flow" includes the model's own parametric knowledge.

3.4 Benchmark Shallowness and the Long-Horizon Evaluation Gap

The benchmarks that currently drive agentic AI research share a common pathology: task completion

is measured in single-session, noise-free environments. AgentBench [11] tasks have median solution lengths of 4–8 steps. WebArena tasks are completed in a single browser session. Even the more demanding OSWorld and τ-Bench evaluations impose time horizons measured in minutes, not hours or days.

Real-world agentic tasks conducting multi-week research projects, managing ongoing software development workflows, coordinating supply chains are categorically different. Such tasks require the agent to maintain goal coherence across interruptions, to update plans in response to new information that invalidates earlier assumptions, and to interact with other agents and human stakeholders whose goals may shift over time. No current benchmark captures these properties.

This evaluation gap creates a systematic bias in the research agenda: methods that improve performance on existing benchmarks are rewarded, even when those methods provide no benefit or actively harm performance on long-horizon tasks. The hypothesis advanced here is that the current trend of reward-hacking in agentic benchmarks (analogous to the well-documented reward hacking in RLHF [24,25]) is already observable in published results, though the field lacks the tools to detect such hacking systematically.

Structural Problem	Current Paradigm Limitation	Proposed Solution
Ephemeral State & Memory Integrity	Context window is a lossy compressor with no verification mechanism. Belief drift compounds across steps. "Lost-in-the-middle" degradation affects long contexts. No write-ahead log or rollback capability.	Three-tier memory hierarchy (EWM/SPM/LTSM) with write-ahead logging. Commit Protocol requires claim verification before SPM writes. Provenance tracking ensures every long-term claim traces to supporting observations. Enables cross-session coherence and auditability.
Informal Tool Interfaces	Tool schemas specify syntax but not semantic preconditions. Side effects not represented in agent context. Error recovery succeeds only 54-71% of the time. Shared-state consistency problem in multi-agent settings.	AgentContract protocol with typed messages containing preconditions (φ_pre), postconditions (φ_post), and capability tokens. Lightweight verifier LLM checks postconditions. Early failure detection prevents error propagation. Structured audit logs enable causal responsibility assignment.
Implicit Trust Inheritance	Trust propagates by default through agent hierarchies. No distinction between "can do" and "authorized to do". Accountability gaps in harmful outputs. Vulnerable to prompt injection via delegation chains.	Trust-Propagation Algebra (TPA) with explicit trust lattice (UNTRUSTED → LOW → MEDIUM → HIGH → VERIFIED). Trust flows monotonically downward via meet operation. Cryptographic revocation certificates. Compositional safety certificates for bounded pipeline behavior.
Benchmark Shallowness	Single-session evaluations (median 4-8 steps). Time horizons of minutes, not days. No evaluation of cross-session coherence or adversarial robustness. Systematic bias toward methods that exploit benchmark artifacts.	LongHorizonAgentBench (LHAB) with 4 task families over 5-session horizons. Intermediate checkpoint evaluation. Adversarial perturbation tracks. Session serialization tests SPM reconstruction. Human-calibrated rubrics (κ ≥ 0.70) for multi-step quality assessment.

Table 1: Structural Problems in Current Agentic AI Systems and Proposed Solutions

Table 1. Summary of structural limitations in current agentic AI systems and corresponding proposed solutions. Each row addresses one of the four core problems identified in Section 3, showing how current paradigm limitations lead to systematic failures and how the proposed research directions (detailed in Section 4) provide architectural remedies.

4. Proposed Research Directions

4.1 Direction 1: Typed Agent Communication Protocols with Verifiable Postconditions

Motivation

The absence of semantic contracts between agents is the most tractable of the structural problems identified above. Unlike memory architecture or trust models, communication protocols are a surface that can be redesigned without modifying the underlying LLM. This work proposes a typed message-passing protocol termed **AgentContract** that wraps every inter-agent communication in a structured envelope containing preconditions, postconditions, and an authorization token.

Technical Sketch

Formally, a message m in AgentContract is a tuple $m = \langle \tau, \varphi\_pre, \varphi\_post, \alpha, c \rangle$, where $\tau \in T$ is a message type drawn from a predefined ontology of task categories; φ_pre and φ_post are first-order logical formulas over the shared world state; α is a capability token (a cryptographically signed set of permitted actions); and c is the natural-language content.

The receiving agent, before acting on the message, invokes a lightweight verification step:

```
function verify_message(m, world_state):
  assert eval(m.phi_pre, world_state) == True
```

action_result = execute(m.c, m.alpha) # bounded by capability token

assert eval(m.phi_post, world_state_after) == True

if assertion fails: return FAILURE, reason

else: return SUCCESS, world_state_after

Postcondition verification can be implemented in two ways: (a) for deterministic operations (file writes, database updates), direct comparison against the world state; (b) for non-deterministic operations (LLM-generated content), by invoking a separate lightweight verifier LLM a model fine-tuned specifically for constraint satisfaction checking, analogous to the reward model in RLHF but operating at the structural rather than preference level.

The message type ontology T is a key design choice. A hierarchy inspired by speech act theory [26,27] and extended for computational agents is proposed: {QUERY, ASSERT, DELEGATE, COMMIT, REVOKE, REPORT_FAILURE}. Each type carries a default precondition schema that agents must instantiate. For example, a DELEGATE message requires the precondition that the delegating agent possesses the capabilities being delegated, and the postcondition that the delegated task is either completed or explicitly failed no silent abandonment.

Expected Advantages and Plausible Failure Modes

The primary advantage is auditability: every inter-agent interaction produces a structured log that can be replayed, analyzed, and used to assign causal responsibility for outcomes. Secondary advantages include early failure detection (a failed postcondition check terminates the erroneous branch before side effects propagate) and compositionality guarantees (if each message satisfies its contract, the pipeline's overall behavior is bounded by the conjunction of contracts).

The primary failure mode is specification overhead. Writing formal pre/postconditions for every task category is labor-intensive, and underspecified conditions may be trivially satisfied (e.g., φ_post = ⊤). This work proposes addressing this through LLM-assisted contract generation: given a task description in natural language, a dedicated contract-generation model proposes φ_pre and φ_post in a constrained formal language, which a human expert then reviews for high-stakes deployments.

Proof-of-Concept Experiments (2026–2027)

1. Implement AgentContract on top of an existing multi-agent framework (e.g., AutoGen 3.0) and measure the rate of silent postcondition violations on a curated set of 100 multi-step tasks currently passing in the AgentBench evaluation.
2. Train a verifier LLM (7B parameter model) on synthetically generated contract-satisfaction examples and evaluate false-positive/negative rates on held-out tasks.
3. Ablation study: compare multi-agent pipeline success rates on adversarially perturbed tasks (injected false observations) with and without postcondition verification.

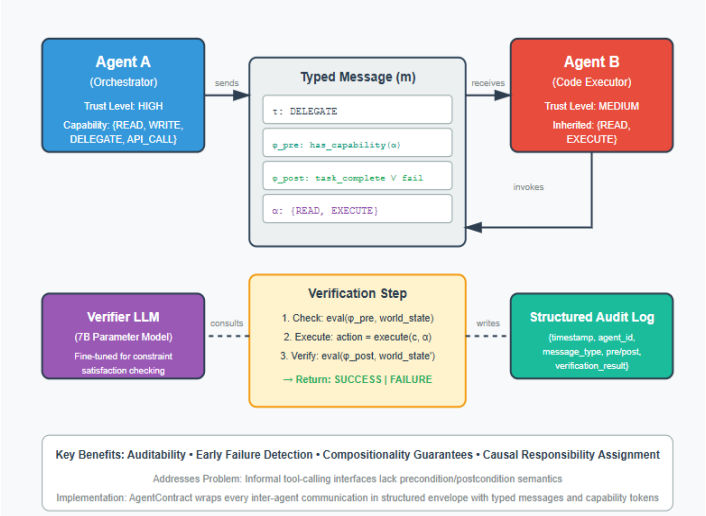

Figure 1: AgentContract Protocol Architecture with Typed Message Passing.

Figure 1 shows the AgentContract protocol architecture for typed inter-agent communication. Agent A (orchestrator) sends a structured message m = ⟨τ, φ_pre, φ_post, α, c⟩ to Agent B (code executor). The verification step evaluates preconditions, executes the action within capability bounds (α), and validates postconditions using a lightweight verifier LLM. All interactions are logged to a structured audit trail, enabling causal responsibility assignment and early failure detection.

4.2 Direction 2: Hierarchical Memory Architectures with Write-Ahead Semantics

Motivation

The memory integrity problem identified in Section 3.1 requires an architectural change, not merely a prompting intervention. This work proposes a three-tier memory hierarchy Ephemeral Working Memory (EWM), Session-Persistent Memory (SPM), and Long-Term Structured Memory (LTSM) with explicit write-ahead logging at the EWM→SPM transition.

Technical Sketch

The architecture is as follows. EWM is the standard context window, unchanged. SPM is a session-scoped key-value store with append-only semantics: the agent can write new entries but cannot overwrite existing ones within a session. LTSM is a persistent graph database (e.g., a property graph) that the agent populates at session end via a structured summarization step.

The write-ahead log (WAL) operates at the EWM→SPM boundary. Before any information from EWM is committed to SPM, the agent executes a Commit Protocol:

```
function commit_to_spm(belief_state, supporting_observations):
    claim = extract_claim(belief_state)          # structured extraction
    evidence = retrieve_supporting(claim, supporting_observations)
    confidence = verify_claim(claim, evidence)   # verifier call
    if confidence >= theta:
        wal.append({claim, evidence, confidence, timestamp})
        spm.write(claim)
    else:
        wal.append({claim, UNVERIFIED, confidence, timestamp})
        # claim enters SPM as flagged-uncertain
```

The threshold θ is a hyperparameter controlling the fidelity/coverage tradeoff. The WAL provides a complete, tamper-evident record of all belief updates, enabling post-hoc auditing of belief drift. The LTSM graph stores entities and relations extracted from SPM summaries; edges carry provenance pointers back to WAL entries, ensuring that every claim in long-term memory is traceable to the observations that supported the claim.

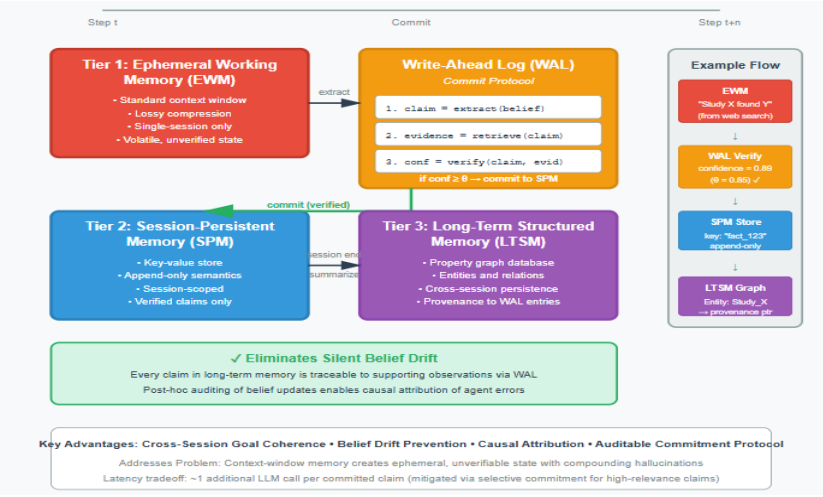

Figure 2: Hierarchical Memory Architecture with Write-Ahead Logging.

Figure 2. Three-tier hierarchical memory architecture with write-ahead logging. Information flows from Ephemeral Working Memory (EWM, context window) through a commit protocol in the Write-Ahead Log (WAL) to Session-Persistent Memory (SPM, key-value store) and finally to Long-Term Structured Memory (LTSM, property graph). The commit protocol verifies claims against evidence with confidence threshold θ before allowing SPM writes, eliminating silent belief drift. All LTSM entries maintain provenance pointers to originating WAL entries, enabling complete auditability of belief updates.

This architecture is inspired by, but distinct from, existing work on retrieval-augmented generation. RAG augments the context with retrieved documents but does not provide write semantics, consistency guarantees, or provenance tracking. The key novelty is the explicit separation of read and write pathways and the grounding of long-term memory in an auditable commitment protocol.

Expected Advantages and Plausible Failure Modes

Advantages: eliminates silent belief drift; enables cross-session goal coherence; supports causal attribution of agent errors to specific belief-update steps. Failure modes: the Commit Protocol adds latency roughly one additional LLM call per committed claim. At high task frequencies (e.g., real-time web agents), this may be prohibitive. The hypothesis is that selective commitment (committing only claims that affect future planning, as predicted by a lightweight relevance classifier) reduces overhead to acceptable levels, though this remains to be empirically validated.

Proof-of-Concept Experiments (2026–2027)

1. Implement the three-tier hierarchy in a research agent prototype and measure belief drift rate on a 100-question fact-tracking benchmark requiring multi-step information synthesis.
2. Compare final-answer accuracy with and without WAL on tasks where ground-truth intermediate beliefs are observable (synthetic tasks with known evidence chains).
3. Measure latency overhead of the Commit Protocol at varying task frequencies and validate the selective commitment hypothesis.

4.3 Direction 3: A Trust-Propagation Algebra for Multi-Agent Delegation

Motivation

The implicit trust inheritance problem (Section 3.3) requires a formal model that makes trust explicit, composable, and revocable. This work proposes a **Trust-Propagation Algebra (TPA)** that assigns a trust level to every agent in a pipeline, computes the trust level of delegated tasks as a function of the delegator's trust and the delegation context, and supports runtime revocation of trust tokens.

Technical Sketch

Trust is modeled as a partially ordered set $(L, \leq)$, where L is a finite lattice of trust levels (e.g., {UNTRUSTED, LOW, MEDIUM, HIGH, VERIFIED}) and $\leq$ is the information-flow ordering. Each agent a_i in a pipeline is assigned a trust level $\lambda(a_i) \in L$. When agent a_i delegates a task to agent a_j, the effective trust level of the delegated task is:

$\lambda(a_j \mid a_i \text{ delegates}) = \lambda(a_i) \sqcap \lambda\_context(delegation\_message)$

where $\sqcap$ is the meet (greatest lower bound) in L, and $\lambda\_context$ is a context-sensitive trust modifier that can reduce but never increase trust based on the semantic content of the delegation message. This monotonic-downward property ensures that trust cannot be amplified through delegation a direct encoding of the principle of least authority.

Trust levels flow downward through the delegation graph. A human operator at the root of the pipeline holds VERIFIED trust. A tool-using sub-agent three levels deep holds at most the meet of the trust levels along the path from root to the sub-agent. Runtime revocation is implemented via a trust revocation certificate, cryptographically signed by a node with authority over the revoked agent, that propagates down the delegation graph and freezes the affected agent's capability tokens.

The algebra also defines a composition operator for task outputs: if agent a_i (trust λ_i) produces an output that is used as input by agent a_j (trust λ_j), then a_j must treat the input as having effective trust $\min(\lambda_i, \lambda_j)$ the agent cannot bootstrap higher-trust conclusions from lower-trust inputs.

Expected Advantages and Plausible Failure Modes

The primary advantage of TPA is that the algebra provides a compositional safety certificate: if every agent in a pipeline behaves correctly with respect to its assigned trust level, the pipeline's outputs carry a provably bounded trust level. This enables risk-stratified deployment: high-stakes actions (financial transactions, system-level commands) can be gated on a minimum trust threshold.

The primary challenge is trust level assignment: determining the appropriate $\lambda(a_i)$ for a given agent requires human judgment or a meta-evaluation process. The proposal is that trust levels be initialized conservatively (all sub-agents at LOW) and promoted through a combination of (a) formal verification of agent behavior on a test suite and (b) runtime track record a Bayesian update rule that increases trust proportionally to the number of correctly executed tasks with verified postconditions, drawing on the AgentContract protocol from Direction 1.

Proof-of-Concept Experiments (2026–2027)

1. Implement TPA in a 3-agent pipeline (orchestrator, retriever, code executor) and verify that prompt injection attacks targeting the retriever cannot propagate HIGH-trust actions to the code executor.
2. Design a red-teaming evaluation where adversarial instructions are injected at each level of the pipeline and measure the containment rate as a function of trust level granularity (coarse 3-level vs. fine 5-level lattice).
3. Evaluate the Bayesian trust promotion mechanism on a set of 50 tasks with known difficulty and measure convergence rate to correct trust levels.

4.4 Direction 4: LongHorizonAgentBench (LHAB) A Benchmark for Long-Horizon Agentic Tasks

Motivation

Existing benchmarks systematically underestimate the difficulty of real agentic tasks by collapsing multi-day processes into single-session evaluations. This work proposes **LongHorizonAgentBench (LHAB)**, a benchmark suite designed to evaluate agent performance over task horizons measured in days, across interrupted sessions, and in multi-agent settings with adversarially perturbed intermediate outputs.

Technical Sketch

LHAB is organized around four task families, each representing a class of real-world long-horizon problems:

1. **ResearchSynth (RS):** The agent must produce a literature synthesis report on a specified topic, requiring retrieval, reading, note-taking, cross-referencing, and writing over a simulated 5-session horizon. Intermediate notes are stored in a simulated SPM that is subject to corruption at randomized intervals.
2. **SoftwareProject (SP):** The agent manages a multi-file software project from specification to tested implementation, with simulated user-feedback interruptions and changing requirements between sessions.
3. **DataPipeline (DP):** The agent builds and validates a data processing pipeline, with ground-truth validation at intermediate checkpoints and injected upstream data quality issues.
4. **MultiAgentNeg (MAN):** Two agent instances with partially misaligned objectives must negotiate a joint plan, with evaluation measuring both task completion and alignment with the human-specified overall goal.

The key methodological innovations in LHAB are:

- **Session serialization:** tasks are split across multiple simulated sessions, and agents are evaluated on the ability to reconstruct context from SPM rather than from live context.
- **Intermediate checkpoint evaluation:** in addition to final outputs, LHAB evaluates the quality of intermediate artifacts (plans, partial implementations, notes) at standardized checkpoints.
- **Adversarial perturbation tracks:** a separate track introduces noise into the information available to the agent (corrupt web search results, hallucinated tool responses) and measures robustness.
- **Human-rater calibration:** all task completions are evaluated against rubrics developed and calibrated by domain experts,

with inter-rater reliability targets of Cohen's $\kappa \geq 0.70$.

Expected Advantages and Plausible Failure Modes

LHAB directly addresses the evaluation gap identified in Section 3.4. The main risk is annotation cost: multi-session task evaluation requires substantially more human effort than single-session scoring. The proposal is to mitigate this through automated rubric application using a dedicated evaluation LLM (separate from the agent being evaluated) for the first-pass scoring, with human review reserved for borderline cases. The evaluation LLM can itself be validated against human annotations on a calibration set, a practice now standard in LLM-as-judge evaluations [25,28].

Proof-of-Concept Experiments (2026–2027)

1. Release a pilot version of LHAB with 50 tasks in each of the RS and SP families, with fully annotated checkpoints and rubrics.
2. Establish baseline results for 3–4 frontier model families and report the performance gap relative to AgentBench on the same underlying tasks.
3. Validate the adversarial perturbation track by measuring the correlation between perturbation severity and performance degradation across model families.

Task Family	Description & Requirements	Evaluation Dimensions
ResearchSynth (RS)	Literature synthesis report over 5 simulated sessions. Requires retrieval, reading, note-taking, cross-referencing, and writing. SPM subject to randomized corruption. Tests multi-session knowledge accumulation and belief consistency. *Target: 5-session horizon with state persistence*	• Final report quality (completeness, accuracy, coherence) • Intermediate note quality at checkpoints • Recovery from SPM corruption events • Citation accuracy and provenance tracking • Cross-session goal maintenance
SoftwareProject (SP)	Multi-file software project from specification to tested implementation. Simulated user feedback interruptions and changing requirements between sessions. Tests adaptive planning and context reconstruction. *Target: Spec-to-deployment with requirement drift*	• Code correctness (test pass rate) • Architecture quality and maintainability • Requirement change adaptation speed • Documentation completeness • Context preservation across interruptions
DataPipeline (DP)	Build and validate data processing pipeline with ground-truth validation at intermediate checkpoints. Injected upstream data quality issues test error detection and recovery. Multi-stage validation gates. *Target: ETL workflow with adversarial data corruption*	• Pipeline correctness (output accuracy) • Error detection rate for data quality issues • Recovery strategy effectiveness • Intermediate checkpoint validation pass rate • Resource efficiency (time, API calls)
MultiAgentNeg (MAN)	Two agent instances with partially misaligned objectives negotiate joint plan. Tests coordination, trust propagation, and alignment preservation. Evaluates both individual agent behavior and system-level outcomes. *Target: Multi-agent coordination with goal conflict*	• Task completion success rate • Alignment with human-specified overall goal • Negotiation efficiency (rounds to convergence) • Trust protocol compliance • Containment of misalignment (safety metrics)

Note: All LHAB task families include adversarial perturbation tracks with injected noise (corrupt web search results, hallucinated tool responses) to measure robustness. Inter-rater reliability target: Cohen's $\kappa \geq 0.70$ across all human-evaluated dimensions.

Table 2: LHAB Task Families and Evaluation Dimensions.

Table 2. Long Horizon Agent Bench (LHAB) task families and evaluation dimensions. Each task family represents a class of real-world long-horizon problems requiring multi-session continuity, intermediate checkpoint validation, and robustness to adversarial perturbations. All tasks include human-calibrated rubrics with inter-rater reliability targets of Cohen's $\kappa \geq 0.70$.

5. Related Work

This work draws on and extends several lines of research. The contributions are positioned relative to the most directly relevant 2024–2026 efforts.

Multi-agent LLM systems. The AutoGen framework [29] and its successors have demonstrated the practical value of structured agent communication, but do not formalize message semantics or provide compositionality guarantees. OpenAgents [30] and MetaGPT [31] similarly focus on capability demonstration rather than formal safety properties. The AgentContract proposal builds on this

engineering work but adds the typed message and postcondition machinery that these frameworks lack.

Agent memory. MemGPT [32] pioneered the explicit management of hierarchical memory in LLM agents, introducing the distinction between main context and external storage. More recent work [33,34] has extended this to multi-session settings. The contribution here is the write-ahead log and the provenance-tracking graph, which address the memory integrity problem that these systems leave open. No prior work is known that formally addresses belief drift in multi-step agents.

Safety and trust in agentic AI. The alignment literature has extensively studied single-turn LLM safety [35,36] but has only recently begun to address agentic settings. R&D Safety [37], the "Constitutional AI for Agents" framework [1], and the concurrent work on agent sandboxing [38] all address aspects of agentic safety. The Trust-Propagation Algebra is closest in spirit to capability-based security models from distributed systems [22] and to information flow control [23], adapted and formalized for the LLM context.

Agentic benchmarks. The most directly related work to LHAB is τ-Bench [14], which introduces more realistic task evaluation through tool-augmented reasoning, and AssistGUI [39], which evaluates GUI-based agents on real applications. Neither addresses multi-session continuity or adversarial intermediate perturbation. LHAB is also related to the recent GAIA benchmark [40], which targets general AI assistant capabilities, but GAIA tasks remain single-session.

6. Discussion and Risks

6.1 Risks of Pursuing the Proposed Path

The most significant risk of formalizing agent communication and trust is a false sense of security. A system with typed messages and trust algebras that passes its verification suite may still exhibit unsafe behavior in distribution-shifted deployments. Formal verification has well-known incompleteness properties, and the semantic gap between natural language task descriptions and formal pre/postconditions will always leave residual ambiguity. The emphasis here is that the proposals in this paper are intended as necessary conditions for safe agentic deployment, not sufficient ones.

A second risk concerns the overhead of the proposed mechanisms. Postcondition verification, write-ahead logging, and trust token management all add latency and complexity. In competitive deployment environments where speed and cost are primary metrics, pressure will exist to disable or simplify these mechanisms. The research community has a responsibility to demonstrate that the safety overhead is acceptable relative to the reduction in failure rates this is an empirical question that the proof-of-concept experiments in Section 4 are designed to begin answering.

6.2 Risks of Ignoring the Proposed Path

The risks of continuing to deploy agentic systems without compositional safety guarantees are, in the assessment presented here, substantially larger. As agentic systems are granted access to more consequential tools, such as financial APIs, code deployment pipelines, and communication systems the expected cost of compositional failures grows. The November 2025 incident described in Section 1 was relatively contained; future incidents in less monitored settings may not be.

A scientific cost also exists to the current absence of rigorous evaluation. Without benchmarks that capture long-horizon behavior, the research community cannot distinguish progress in agentic capability from progress in benchmark gaming. The risk exists of repeating the NLP evaluation crisis of 2019–2021 [41,42], where models achieving human-level performance on benchmarks were found to rely on spurious correlations. LHAB is intended in part to preempt this failure mode in agentic research.

6.3 Limitations of the Proposals

Three significant limitations are acknowledged. First, the AgentContract proposal assumes that tasks can be partially formalized in a first-order logic fragment this assumption is reasonable for structured, tool-using tasks but may not hold for open-ended creative or social tasks. Second, the Trust-Propagation Algebra assumes a hierarchical delegation structure; peer-to-peer multi-agent coordination (e.g., debate or voting protocols) requires a different formal treatment that has not been developed here. Third, LHAB requires domain-expert annotation that is expensive and potentially inconsistent across domains; scaling the benchmark across diverse task types will require investment in annotation

infrastructure that exceeds what a single research group can provide.

6.4 Open Problems

Several important problems are outside the scope of this paper. The question of how to specify agent goals in a way that is simultaneously expressive enough to capture human intent and constrained enough to permit formal verification sometimes called the specification problem is arguably the deepest open question in alignment and is not addressed here. The interaction between the proposed trust model and adversarial attacks on the LLM itself (as opposed to the pipeline) is also unexplored. Finally, the societal implications of highly autonomous multi-agent systems accountability gaps, labor displacement, novel attack surfaces for state-level adversaries deserve sustained attention from policy researchers in parallel with the technical work proposed here.

7. Conclusion

Agentic AI systems are not simply scaled-up chatbots. Such systems are distributed programs that operate over extended time horizons, invoke tools with real-world side effects, and delegate subtasks to other autonomous systems. The failure taxonomy of distributed programs is well understood; the failure taxonomy of multi-agent LLM pipelines is not. This paper has argued that this gap is the central bottleneck in agentic AI research, not capability but compositionality.

Four directions have been proposed: typed communication protocols with verifiable postconditions, hierarchical memory with write-ahead semantics, a trust-propagation algebra for delegation chains, and a long-horizon benchmark family that together constitute a tractable research agenda for the next 36 months. Each direction is technically specified, experimentally falsifiable, and grounded in prior theory from adjacent fields (distributed systems, formal verification, information flow security) that the AI community has not yet fully assimilated.

The field has a choice. Increasingly capable agents can continue to be built on the informal foundations of the current paradigm, discovering compositional failure modes empirically in deployment and patching such modes reactively. Or investment can occur now in the formal infrastructure that would make agentic AI systems both more capable and more trustworthy. The belief here is that the second path is both technically feasible and empirically necessary.

The most capable AI agent is the one that can be trusted to act on behalf of users when those users are not watching. Earning that trust requires a theory of delegation, and such a theory does not yet exist

References

[1] WILLIAM TORGBI AGBEMABIESE, Toward Constitutional Autonomy in AI Systems: A Theoretical Framework for Aligned Agentic Intelligence. IEEE Xplore, 2025. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=11354471 .

[2] Anthropic, Claude's Character, and Agentic Capabilities: Technical Report on Claude 3.7 Sonnet. Technical Report, Anthropic, 2025. https://www.anthropic.com/news/claude-3-7-sonnet

[3] Google DeepMind, Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next-Generation Agentic Capabilities. Google DeepMind, 2026. https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf

[4] Nathan Schlaffer, Cobus Greyling. Parallel Agent Processing, 2025. https://www.kore.ai/ai-insights/parallel-agent-processing

[5] Joon Sung Park et al., Social Simulacra: Creating Populated Prototypes for Social Computing Systems. ACM Digital Library, 2025. https://dl.acm.org/doi/10.1145/3526113.3545616

[6] Gheorghe Comanici et al. “Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities.” Technical Report, Google DeepMind, 2025. https://arxiv.org/abs/2507.06261

[7] Anthropic, Scaling Long-Context Reasoning in Claude 4. Technical Report, Anthropic, 2025. https://www.anthropic.com/news/claude-sonnet-4-6

[8] Fábio Perez, Ian Ribeiro, Ignore Previous Prompt: Attack Techniques for Language Models. In Proc. NeurIPS ML Safety Workshop 2022. https://arxiv.org/abs/2211.09527

[9] Kai Greshake et al., "Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection." arXiv:2302.12173 [cs.CR] 2023. https://arxiv.org/abs/2302.12173

[10] Yangjun Ruan et al., Identifying the Risks of LM Agents with an LM-Emulated Sandbox. arXiv:2309.15817 [cs.AI], 2024. https://arxiv.org/abs/2309.15817

[11] Xiao Liu et al., AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688 [cs.AI], 2025. https://arxiv.org/abs/2308.03688

[12] Shuyan Zhou et al., WebArena: A Realistic Web Environment for Building Autonomous Agents. arXiv:2307.13854 [cs.AI], 2024. https://arxiv.org/abs/2307.13854

[13] Tianbao Xie et al., OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. arXiv:2404.07972 [cs.AI], 2024. https://arxiv.org/abs/2404.07972

[14] Shunyu Yao et al., τ-Bench: A Benchmark for Tool-Augmented Language Agent Evaluation in Real-World Domains. arXiv:2406.12045 [cs.AI], 2025. https://arxiv.org/abs/2406.12045

[15] Freda Shi et al., Large Language Models Can Be Easily Distracted by Irrelevant Context. ACM Digital Library, 2023. https://arxiv.org/abs/2302.00093

[16] Cheng-Ping Hsieh et al., RULER: What's the Real Context Size of Your Long-Context Language Models? arXiv:2404.06654 [cs.CL], 2024.https://arxiv.org/abs/2404.06654

[17] Nelson F. Liu et al., Lost in the Middle: How Language Models Use Long Contexts Transactions of the ACL, arXiv:2307.03172 [cs.CL], 2023. https://arxiv.org/abs/2307.03172

[18] Yejin Bang et al., A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In Proc. AACL 2023.https://arxiv.org/abs/2302.04023

[19] Lei Huang et al., A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Digital Library, 2025. https://dl.acm.org/doi/10.1145/3703155

[20] John Yang et al., InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback. arXiv:2306.14898 [cs.CL], 2024. https://arxiv.org/abs/2306.14898

[21] Qiusi Zhan, Zhixiang Liang, InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. arXiv:2403.02691 [cs.CL], 2024. https://arxiv.org/abs/2403.02691

[22] Mark S. Miller et al., Capability Myths Demolished. Technical Report, Johns Hopkins University Systems Research Laboratory, 2003. https://classpages.cselabs.umn.edu/Fall-2021/csci5271/papers/SRL2003-02.pdf

[23] A. Sabelfeld and A.C. Myers, Language-Based Information-Flow Security. IEEE Xplore, 2003. https://ieeexplore.ieee.org/document/1159651

[24] Leo Gao et al., “Scaling Laws for Reward Model Overoptimization.” In Proc. ICML 2023. https://proceedings.mlr.press/v202/gao23h/gao23h.pdf

[25] Yann Dubois et al., “Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators.” In Proc. ACL 2024. https://arxiv.org/abs/2404.04475

[26] Austin, JL, How to Do Things with Words. Oxford University Press. 1962. https://silverbronzo.wordpress.com/wp-content/uploads/2017/10/austin-how-to-do-things-with-words-1962.pdf

[27] Searle, John R, Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, 1969. https://archive.org/details/speechactsessayi0000sear

[28] Lianmin Zheng et al., Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL], 2023. https://arxiv.org/abs/2306.05685

[29] Qingyun Wu et al., AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155 [cs.AI], 2023. https://arxiv.org/abs/2308.08155

[30] Tianbao Xie et al., OpenAgents: An Open Platform for Language Agents in the Wild. arXiv:2310.10634 [cs.CL], 2023. https://arxiv.org/abs/2310.10634

[31] Sirui Hong et al., MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv preprint arXiv:2308.00352. 2023. https://arxiv.org/abs/2308.00352

[32] Charles Packer et al., MemGPT: Towards LLMs as Operating Systems.

arXiv:2310.08560 [cs.AI], 2024. https://arxiv.org/abs/2310.08560

[33] Wanjun Zhong et al., MemoryBank: Enhancing Large Language Models with Long-Term Memory. Proceedings of the AAAI Conference on Artificial Intelligence, 2024. https://ojs.aaai.org/index.php/AAAI/article/view/29946

[34] Fouad Bousetouane, "AI Agents Need Memory Control Over More Context." arXiv:2601.11653 [q-bio.NC], 2026. https://arxiv.org/abs/2601.11653

[35] Yuntao Bai et al., "Training a helpful and harmless assistant with reinforcement learning from human feedback." arXiv preprint arXiv:2204.05862, 2022. https://arxiv.org/abs/2204.05862

[36] Long Ouyang et al., Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL], 2022. https://arxiv.org/abs/2203.02155

[37] Anthropic, "Responsible Scaling Policy: Frontier AI Safety Commitments." Technical Report, Anthropic, 2024. https://assets.anthropic.com/m/24a47b00f10301cd/original/Anthropic-Responsible-Scaling-Policy-2024-10-15.pdf

[38] Ivan Nardini. "Introducing Code Execution: The code sandbox for your agents on Vertex AI Agent Engine." In Proc. NDSS 2025. https://discuss.google.dev/t/introducing-code-execution-the-code-sandbox-for-your-agents-on-vertex-ai-agent-engine/264336

[39] Difei Gao et al., AssistGUI: Task-Oriented Desktop Graphical User Interface Automation. In Proc. CVPR 2024. https://arxiv.org/html/2312.13108v2

[40] Grégoire Mialon et al., GAIA: A Benchmark for General AI Assistants. arXiv preprint arXiv:2311.12983, 2023. https://arxiv.org/abs/2311.12983

[41] Suchin Gururangan et al., Annotation Artifacts in Natural Language Inference Data. In Proc. NAACL 2018. https://aclanthology.org/N18-2017/

[42] R. Thomas McCoy et al., Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Proc. ACL 2019. https://aclanthology.org/P19-1334/