Beyond Text-Passing: Shared Cognitive Substrates for Multi-Agent LLM Coordination

Ning Coeva
Independent Researcher
coevaxlumen.log@gmail.com

Abstract

Current multi-agent LLM systems rely on natural language as the sole medium for inter-agent communication, treating coordination as a message-passing problem. We argue this text-passing bottleneck is a structural limitation—not merely an engineering inconvenience—that fundamentally constrains consistency, efficiency, and auditability. We propose a paradigm shift: from message-passing to shared cognitive substrates, where agents coordinate through three explicit shared primitives: (1) a Shared World Model maintaining typed state with invariants, (2) a Shared Causal Graph representing explicit dependencies and attribution paths, and (3) a Shared Energy Pool enabling resource-aware arbitration under budget constraints. We introduce Baton Arbitration as a coordination mechanism and propose Structural Continuity as an evaluation criterion for system stability. We present this as a blueprint intended to guide the design of substrate-native multi-agent systems.

1 Introduction

The emergence of multi-agent LLM systems has enabled increasingly sophisticated coordination patterns, from role-based dialogue to complex workflow orchestration (Wu et al., 2023; LangChain, 2024). Yet beneath the diversity of frameworks lies a common assumption: natural language is the primary medium of inter-agent state. This assumption persists despite decades of multi-agent systems research demonstrating that effective coordination requires explicit shared representations (Durfee, 1999; Wooldridge, 2009).

We challenge this assumption. When Agent A summarizes its reasoning for Agent B in natural language, information is inevitably compressed, types are lost, and consistency becomes unverifiable. When multiple agents write to a shared narrative, conflicts are resolved through generation rather than arbitration. When coordination scales, token bandwidth becomes system bandwidth.

Thesis. Natural language should serve as an interface to multi-agent coordination, not its substrate. We propose that effective multi-agent systems require explicit shared structures—what we term shared cognitive substrates—that maintain state, causality, and resource allocation independently of linguistic representation. This echoes Minsky's vision of intelligence as a society of specialized agents (Minsky, 1986), but shifts the focus from agent-internal architecture to the inter-agent coordination medium itself.

Contributions. This paper offers three contributions: (1) A bottleneck taxonomy identifying three failure modes of text-passing architectures. (2) A substrate abstraction proposing three shared primitives (World Model, Causal Graph, Energy Pool) as coordination foundations. (3) A coordination-evaluation framework introducing Baton Arbitration for conflict-free execution and Structural Continuity as a stability metric.

2 The Text-Passing Bottleneck

2.1 Positioning: What Shared Substrates Are Not

Before detailing our proposal, we clarify what shared cognitive substrates are not:

- Not a shared prompt or memory bank. Substrates are typed, schema-bound structures with mutation interfaces—not text buffers that agents read and append to.
- Not a message queue or cache. We do not merely store messages for later retrieval; we maintain world state that agents query and mutate through controlled operations.
- Not generic concurrency control. While coordination involves access management, substrates provide semantic primitives—typed state with invariants, causal provenance with propagation, and budget-aware arbitration—rather than low-level locks or queues.

The core distinction: we define coordination primitives at the cognitive level, not communication tricks at the infrastructure level. Unlike classic blackboard architectures (Hayes-Roth, 1985), substrates here are explicitly tied to causal provenance and budgeted arbitration, making multi-agent coordination measurable and auditable.

2.2 Failure Mode Taxonomy

Existing multi-agent LLM frameworks—including AutoGen (Wu et al., 2023), LangGraph (LangChain, 2024), CrewAI (CrewAI, 2024), CAMEL (Li et al., 2023), MetaGPT (Hong et al., 2023), ChatDev (Qian et al., 2023), GPTSwarm (Zhuge et al., 2024), and simulation environments such as Generative Agents (Park et al., 2023)—share a fundamental architectural choice: agents communicate by generating and consuming natural language messages. While some frameworks introduce structured artifacts (e.g., MetaGPT's SOPs) or role-based protocols (e.g., CAMEL's inception prompting), the inter-agent medium remains linguistic. We identify three structural failure modes inherent to this design.

Information Loss. When an agent externalizes internal state as text, typed structures become untyped strings. A causal chain $A \to B \to C$ becomes "A leads to B, which causes C"—losing explicit edge semantics, confidence annotations, and compositional structure. Downstream agents must re-infer what upstream agents already knew.

Consistency Failure. Without explicit dependency tracking, parallel agents may hold contradictory beliefs about shared state. Text-based reconciliation requires generation ("let me clarify...") rather than verification against invariants. There is no mechanism to propagate updates or detect when Agent A's conclusions invalidate Agent B's premises.

Coordination Overhead. Communication cost scales with token count, not semantic content. Expressing "Agent 3 should proceed" requires the same bandwidth machinery as complex reasoning. As agent count and interaction depth increase, coordination overhead dominates useful computation.

Failure Mode	Root Cause	Required Primitive(s)
Information Loss	Untyped serialization	World Model (typed state)
Consistency Failure	No dependency tracking	World Model + Causal Graph
Coordination Overhead	Token = bandwidth	Energy Pool (budget arbitration)

Table 1: Text-passing failure modes and their corresponding substrate primitives.

3 Shared Cognitive Substrates

We propose that multi-agent coordination should be grounded in explicit shared structures rather than emergent from message exchange. A shared cognitive substrate provides three primitives that agents read from, write to, and reason over collectively.

Shared World Model (State). The Shared World Model maintains a typed representation of entities, relations, attributes, and constraints that all agents observe and modify through controlled interfaces. Core properties: typed state (objects carry schemas; relations have defined semantics), invariant enforcement (constraints are checked on write, not hoped for on read), and versioned history (state transitions are logged, enabling rollback and audit). Agents do not "describe" the world to each other; they observe the same world.

Shared Causal Graph (Structure). The Shared Causal Graph explicitly represents dependencies, derivations, and attribution paths across agent reasoning. Core properties: explicit causality (if conclusion C depends on premises P_1, P_2, edges $P_1 \rightarrow C$ and $P_2 \rightarrow C$ are recorded), cross-agent attribution (when Agent B builds on Agent A's output, the dependency is structural, not textual), and consistency propagation (updating a node triggers downstream validity checks). This enables credit assignment across agents, contradiction detection, and principled belief revision.

Shared Energy Pool (Resource). The Shared Energy Pool provides a resource-aware coordination layer where agents operate under explicit budget constraints. Core properties: finite budget (total available "energy" is bounded; agents request allocations), priority arbitration (when demand exceeds supply, allocation follows explicit rules), and contention visibility (resource conflicts are observable events). Here, "energy" is an abstract budget that can be instantiated as operation quota, validation quota, or tool-call budget depending on deployment context. Crucially, energy allocations gate which substrate operations can be executed under contention: when budget is tight, agents must trade off between proposing, committing, validating, and repairing state—making coordination an explicit, auditable control surface rather than an implicit latency.

Walkthrough: Multi-Agent Planning. This walkthrough illustrates coordination in terms of substrate operations, decoupled from message length and linguistic reconciliation.

Example: Two Agents Collaboratively Refine a Plan

Setup: Agent A (Planner) and Agent B (Critic) collaborate on a deployment plan.
World Model writes: A writes step1: "provision servers" [status=proposed]; A writes step2: "deploy code" [depends=step1, status=proposed].
Causal Graph edges: step1 → step2 (dependency); Agent_A → step1, step2 (provenance).
Energy arbitration: Budget allows 3 more operations. B requests 2 units for critique; granted.
Baton transfer: A holds write-baton, commits steps. A releases baton. B acquires baton, writes: step1.risk: "capacity unknown". Causal graph adds: Agent_B → step1.risk.
Consistency propagation: Risk annotation triggers re-validation flag on step2.

In a text-passing architecture, the same update would require repeated natural-language reconciliation to re-establish dependencies and resolve conflicts, inflating coordination cost without increasing semantic precision.

4 Baton Arbitration: A Coordination Primitive

Given shared substrates, how should agents coordinate access? Classical multi-agent coordination relies on negotiation protocols, shared plans, or learned policies (Stone & Veloso, 2000). We propose a simpler primitive suited to substrate-native systems: Baton Arbitration, where execution rights are explicitly held and transferred, rather than implicitly contested.

The Baton Concept. A baton represents the right to perform a privileged operation on shared state—writing to the World Model, committing to the Causal Graph, or allocating from the Energy Pool. At any moment, each baton has exactly one holder. Benefits: conflict elimination (writers don't race; they wait for the baton), responsibility attribution (every state change has an accountable agent), and rollback simplicity (reverting a bad commit means returning to pre-baton state).

Batons are not generic mutexes. Unlike low-level locks, a baton is a semantic write authority bound to substrate regions, with logged commits for audit and role-based partitioning. Acquiring a baton implies accepting accountability for the coherence of subsequent

state changes—not merely exclusive access to a resource. Importantly, batons govern only privileged writes and commits; reads and queries can proceed concurrently without baton acquisition, and separate batons can be defined per substrate region (world-state, causal, budget) to enable fine-grained parallelism.

Role-Based Partitioning. Rather than treating all agents as interchangeable, we suggest role-based capability partitioning: different roles hold authority over different substrate regions. Illustrative roles: World Editor (authorized to mutate entity/relation state), Causal Analyst (authorized to extend/prune the causal graph), Energy Governor (authorized to reallocate budget across agents), and Continuity Monitor (authorized to trigger stability interventions). This transforms coordination from "who talks next" to "who holds write authority now."

5 Structural Continuity: An Evaluation Criterion

How do we know if a multi-agent system is "working well"? Beyond task completion, we propose Structural Continuity as a stability metric.

Structural Continuity measures the degree to which a multi-agent system maintains coherent shared state across interactions, defined along three dimensions: State Stability (how much does the World Model drift under agent activity?), Causal Integrity (are causal graph edges preserved or frequently invalidated?), and Energy Efficiency (what fraction of allocated resources produces durable changes, i.e., changes surviving k steps without invalidation?).

Formally, over a horizon k, we view structural continuity as $\mathrm{SC}_k = f(\Delta_{\text{state}}, \rho_{\text{causal}}, \eta_{\text{energy}})$, where Δ_{state} captures world-model drift, ρ_{causal} denotes causal-edge survival rate, and η_{energy} denotes the fraction of budget yielding durable commits. Specific instantiations of f are left to future empirical work.

Measurement Approach. We suggest continuity can be operationalized through: anchor tracking (designated reference points in shared state; continuity = anchor survival rate over n interactions), drift detection (measuring distance between World Model snapshots at t and $t + k$, via schema-aware state differencing or embedding-based comparison), and recovery cost (resources required to restore consistency after perturbation, measured in energy units or operation count).

6 Implications and Conclusion

Safety. Shared substrates make multi-agent behavior auditable by design. Every state change traces to an agent holding a baton; every causal claim has explicit provenance. This transforms safety from "hoping agents behave" to "verifying agents complied."

Scalability. When coordination occurs through substrate operations rather than token exchange, communication cost decouples from linguistic complexity. Adding agents increases substrate load, not message explosion.

Open Questions. Several challenges remain: What is the minimal substrate sufficient for coherent coordination? Can baton transfer policies be learned rather than predefined? How should substrate-based and text-based coordination interoperate? As a minimal next step, we plan a validation stress suite that systematically varies contention (budget), conflict frequency, and horizon length k, and reports structural continuity (SC_k) together with coordination overhead against text-passing baselines.

We have argued that multi-agent LLM coordination suffers from a structural bottleneck: reliance on natural language as the sole inter-agent medium. We proposed shared cognitive substrates—World Model, Causal Graph, and Energy Pool—as explicit coordination primitives, introduced Baton Arbitration as a conflict-free coordination mechanism, and suggested Structural Continuity as an evaluation criterion. This blueprint aims to shift the conversation from "how agents talk to each other" to "what agents share with each other."

Acknowledgments

LLM Usage Disclosure: All research ideas, theoretical frameworks, and technical content in this paper are solely the original intellectual work of the authors. An LLM-based assistant was used for language refinement and formatting under human oversight.

References

CrewAI. Crewai: Framework for orchestrating role-playing autonomous ai agents. https://github.com/joaomdmoura/crewAI, 2024.

Edmund H Durfee. Distributed problem solving and planning. In Multi-Agent Systems and Applications, pp. 118–149. Springer, 1999.

Barbara Hayes-Roth. A blackboard architecture for control. Artificial Intelligence, 26(3): 251–321, 1985.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. arXiv preprint arXiv:2308.00352, 2023.

LangChain. Langgraph: Build stateful multi-actor applications with llms. https://github.com/langchain-ai/langgraph, 2024.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In Advances in Neural Information Processing Systems, volume 36, 2023.

Marvin Minsky. The Society of Mind. Simon & Schuster, 1986.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. arXiv preprint arXiv:2304.03442, 2023.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. arXiv preprint arXiv:2307.07924, 2023.

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. Autonomous Robots, 8(3):345–383, 2000.

Michael Wooldridge. An Introduction to Multiagent Systems. John Wiley & Sons, 2nd edition, 2009.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. arXiv preprint arXiv:2308.08155, 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. arXiv preprint arXiv:2402.16823, 2024.