Coordination of Governed Agents Over Shared Context: Declarative Governance and Lyapunov-Based Finality

Submitted to arXiv

https://github.com/DealExMachina/swarm-of-governed-agents

February 22, 2026

Abstract

The emergence of LLM-powered autonomous agents has exposed the limits of pipeline-based coordination: rigid DAGs cannot accommodate contradiction, evolving evidence, or formal convergence. We propose *declarative governance over shared immutable context*, a paradigm where autonomous agents reason over a common semantic graph, propose state transitions, and reach consensus through formal convergence mechanisms—without predefined execution sequences. Three innovations enable this: (1) an append-only event log and CRDT-inspired semantic graph providing shared, monotonic context; (2) YAML-defined governance rules as coordination primitives, separating policy from agent implementation; (3) Lyapunov-based finality tracking with monotonicity gates, plateau detection, pressure-directed activation, and human-in-the-loop routing. Fine-grained access control (OpenFGA) governs agent execution, document access, and policy modification. Unlike orchestration frameworks (AutoGen, CrewAI, LangGraph) that wire agents into chains or graphs, our system lets agents independently reason over shared state and propose actions evaluated by declarative policy. Validation on a realistic M&A due-diligence scenario (Project Horizon) demonstrates convergence in 12 rounds across 5 contradictory documents, with 83% autonomous contradiction resolution, complete audit trail, and structured human review of residual disagreements.

1 Introduction

The rapid maturation of LLM-powered agents—capable of reasoning, planning, and tool use—has created a coordination problem that traditional workflow engines were not designed to solve. Modern agents do not merely execute predefined tasks; they interpret context, generate hypotheses, and produce claims that may contradict each other. This capability demands a fundamentally different coordination model.

Current approaches fall into two categories, both inadequate:

Pipeline orchestration (Airflow, Temporal, Prefect) assumes static task graphs executed in topological order. When agents produce contradictory outputs, the pipeline either silently overwrites earlier results or fails entirely. There is no mechanism for tracking disagreement, no formal convergence guarantee, and no governance trail explaining *why* a particular conclusion was reached.

Agent frameworks (AutoGen, CrewAI, LangGraph) represent progress: they enable multi-agent conversations, role-based delegation, and tool use. However, they still wire agents into predefined topologies—sequential chains, hierarchical delegation, or cyclic graphs with hardcoded routing. Coordination logic is embedded in code, not governed by policy. There is no shared immutable context, no formal finality mechanism, and no fine-grained access control.

We propose a third path: *declarative governance over shared context*. Rather than sequencing agents, the system maintains:

1. **Shared immutable context:** An append-only event log and a CRDT-inspired semantic graph where confidence scores ratchet upward, contradictions are tracked explicitly, and no state is ever deleted—only superseded.

2. **Declarative governance:** Agent activation and state transitions are governed by human-readable YAML rules. No agent directly modifies shared state; all propose transitions that governance evaluates before an executor implements them.

3. **Formal convergence:** A Lyapunov disagreement function $V(t)$ tracks distance to targets across four weighted dimensions. Monotonicity gates, plateau detection, and divergence escalation provide formal guarantees. When autonomous convergence stalls, the system routes to structured human review with full context.

4. **Fine-grained access governance:** OpenFGA enforces who can read documents, propose transitions, approve decisions, and modify policies.

This paper presents the design, formal properties, and experimental validation of this system.

2 Related Work

2.1 Pipeline Orchestration

Workflow engines (Apache Airflow, Temporal, Prefect) implement the DAG model: define a static graph, schedule tasks in topological order, apply retry logic on failure. This model is effective for deterministic ETL pipelines but breaks when agents produce conflicting outputs, when new evidence appears mid-execution, or when tasks are interdependent in non-linear ways. None provide formal finality guarantees or governance-as-coordination.

2.2 Agent Coordination Frameworks

Recent frameworks enable multi-agent collaboration:

AutoGen [9] introduces conversable agents with role-based chat patterns. Coordination is conversation-driven: agents exchange messages in predefined patterns (sequential, group chat, nested). However, coordination topology is hardcoded in Python, there is no shared persistent state between conversations, and no formal convergence mechanism.

CrewAI [6] organizes agents into crews with roles, goals, and tools. A hierarchical manager delegates tasks. Coordination follows a fixed process (sequential or hierarchical), with no mechanism for handling contradictions between agent outputs or tracking disagreement over time.

LangGraph [5] models agent coordination as state machines with explicit edges and conditional routing. This is closer to our approach—state is explicit and transitions are defined—but routing logic is embedded in code, not declarative policy. There is no immutable audit log, no CRDT semantics, and no formal convergence tracking.

All three frameworks assume that coordination topology can be predefined. Our system replaces topology with policy: agents reason independently over shared state, propose transitions, and governance rules determine what happens next.

2.3 Consensus and Convergence Theory

Olfati-Saber and Murray [7] established the Lyapunov consensus framework for multi-agent systems, proving that monotonically decreasing disagreement functions guarantee convergence under mild assumptions. We adapt this to knowledge-work coordination, defining a weighted disagreement function over four semantic dimensions.

Duan et al. [3] introduced the Aegean protocol with monotonicity gates and coordination invariants, requiring β consecutive non-decreasing rounds before declaring convergence. We adopt this mechanism directly.

Camacho et al. [1] proposed EMA-based stagnation detection in the MACI framework, identifying when systems plateau despite appearing active. Our plateau detection mechanism derives from this work.

2.4 CRDT Semantics and Monotonic State

Laddad et al. [4] demonstrated CRDT monotonic merge operations (CodeCRDT) that prevent state regression in distributed systems. Our semantic graph adopts this principle: confidence scores ratchet upward only, contradictions once marked are irreversible, and nodes are staled rather than deleted.

2.5 Stigmergic Coordination

Dorigo et al. [2] formalized stigmergic coordination in swarm intelligence: agents respond to environmental signals (pheromones) rather than direct communication. Our pressure-directed activation mechanism applies this principle—agents are routed toward high-pressure dimensions (bottleneck areas) without centralized scheduling.

2.6 Relationship-Based Access Control

Pang et al. [8] formalized Zanzibar-style relationship-based access control, implemented in OpenFGA. We integrate this for fine-grained governance of document access, transition approval, and

policy modification.

3 Core Design

3.1 Design Principle: Proposals, Approval, Execution

The central invariant of the system is: *no agent directly modifies shared state.* Instead:

1. **Agents propose:** Each agent reads shared context, reasons about it, and emits a proposal (a candidate state transition with justification).
2. **Governance evaluates:** Declarative rules (YAML) determine whether the proposal is approved, blocked, or routed to human review.
3. **Executor implements:** Only approved proposals are applied to shared state, via atomic epoch-based compare-and-swap (CAS) transitions.

This pattern provides complete auditability: every state change has a proposal, an evaluation, and an execution record. It also enables governance without code changes: rules are configuration, not compiled logic.

3.2 Shared Immutable Context

All agents read from and write to two shared data structures:

Append-Only Event Log (Context WAL). Every claim, fact, and agent decision is recorded as an immutable event in a Postgres write-ahead log:

```
{
  "timestamp": "2025-08-15T10:32:00Z",
  "agent_id": "facts_extraction_v2",
  "event_type": "claim_emitted",
  "payload": {
    "entity": "NovaTech AG",
    "relation": "annual_recurring_revenue",
    "value": "EUR 50M",
    "confidence": 0.92,
    "source_doc": "analyst_briefing.pdf"
  }
}
```

Events are never modified or deleted. This provides tamper-proof auditability, full reproducibility (replay the log to reconstruct any state), and causal traceability (which facts triggered which agent actions).

CRDT-Inspired Semantic Graph. A Postgres + pgvector database maintains semantic relationships with monotonic guarantees [4]:

- **Monotonic confidence:** Confidence scores ratchet upward only. A claim at 0.85 can become 0.92 but never revert to 0.80.
- **Irreversible contradictions:** Once a contradiction edge is created between two claims, it cannot be deleted—only resolved by creating a resolution edge.
- **Staling, not deletion:** Superseded nodes are marked stale, preserving the full reasoning history.

These CRDT semantics prevent state regression and enable formal convergence guarantees: the semantic graph can only move toward resolution, never away from it.

3.3 Declarative Governance

Agent activation and state transitions are defined in YAML:

```
governance:
  approval_modes:
    YOLO:    # Automatic approval for valid transitions
    MITL:    # All proposals route to human review
    MASTER:  # Deterministic rule-based, no LLM rationale

  drift_rules:
    - type: contradiction
      severity: [medium, high]
      action: open_investigation
    - type: goal_misalignment
      severity: [medium, high]
      action: request_goal_refresh
    - type: entropy
      severity: high
      action: halt_and_review

  transitions:
    - from: DriftChecked
      to: ContextIngested
      blocking_condition: critical_drift
      override: requires_human_authorization
```

Three governance modes provide different compliance profiles:

- **YOLO:** Automatic approval for valid transitions. Suitable for low-risk, internal workflows.

- **MITL (Man-in-the-Loop):** All proposals route to human review queue. Recommended for standard compliance workflows.
- **MASTER:** Deterministic rule-based decisions without LLM rationale. Suitable for highest-sensitivity scenarios.

Per-scope overrides allow mixing modes within a single system (e.g., YOLO for low-risk extraction, MITL for financial claims).

3.4 Three-Node State Cycle

Rather than an open-ended state machine, the system operates on a tight three-node cycle with epoch-based CAS:

$$\texttt{ContextIngested} \rightarrow \texttt{FactsExtracted} \rightarrow \texttt{DriftChecked} \rightarrow \texttt{ContextIngested}$$

Each transition requires governance approval. Only one agent succeeds per cycle (atomic epoch check). This prevents race conditions and ensures every cycle produces a complete audit record.

3.5 Fine-Grained Access Governance (OpenFGA)

OpenFGA [8] enforces relationship-based access control at four levels:

- **Document access:** Only authorized agents and users read sensitive materials.
- **Transition approval:** Governance rules determine who can approve which transitions.
- **Policy modification:** Changes to governance rules require role-based sign-off and version control.
- **Audit access:** Sensitivity-appropriate visibility into audit logs.

4 Convergence Theory and Finality

4.1 The Finality Problem

Traditional systems declare finality by threshold: "if confidence > 0.95, done." This is brittle: contradictions may emerge after the threshold is crossed, the system may cycle indefinitely, and there is no formal guarantee of stabilization. We replace threshold-based finality with *stateful convergence tracking*.

4.2 Lyapunov Disagreement Function

Following Olfati-Saber and Murray [7], we define a scalar disagreement function $V(t) \geq 0$ measuring distance to convergence targets:

$$V(t) = \sum_{d \in \mathcal{D}} w_d \cdot (\text{target}_d - \text{actual}_d(t))^2$$

where $\mathcal{D}$ comprises four weighted dimensions:

Dimension	Target	Weight	Meaning
Claim confidence	≥ 0.85	0.30	Active claims at sufficient confidence
Contradiction resolution	$= 0$	0.30	Zero unresolved contradictions
Goal completion	≥ 0.90	0.25	Completion ratio of declared goals
Risk score inverse	< 0.20	0.15	Residual risk below threshold

Table 1: Convergence dimensions with weights and targets.

4.3 Convergence Rate and ETA

The convergence rate α is computed as:

$$\alpha = -\ln\left(\frac{V(t)}{V(t-1)}\right)$$

with estimated time to arrival:

$$\text{ETA} = \left\lceil \frac{-\ln(\epsilon / V(t))}{\alpha} \right\rceil, \quad \epsilon = 0.005$$

Three regimes:

- $\alpha > 0$: Converging. Finite ETA.
- $\alpha \approx 0$: Stalled. Plateau detection activates.
- $\alpha < -0.05$: Diverging. Triggers ESCALATED state.

4.4 Finality Mechanisms

Monotonicity Gate [3]. Goal score must remain non-decreasing for $\beta = 3$ consecutive rounds before auto-finality eligibility. A single drop > 0.001 resets the counter.

Plateau Detection [1]. Uses EMA of progress ratio:

$$\text{EMA}(t) = 0.3 \cdot \text{progress\_ratio}(t) + 0.7 \cdot \text{EMA}(t-1)$$

Plateau declared when EMA < 0.01 for $\tau = 3$ consecutive rounds.

Pressure-Directed Activation [2]. Agents are routed toward the dimension with highest residual gap (bottleneck), without centralized scheduling. This stigmergic mechanism ensures resources concentrate where they matter most.

Divergence Escalation. If $\alpha < -0.05$ (disagreement increasing), the system transitions to ESCALATED state, requiring human intervention.

Human-in-the-Loop Review. When goal score is in $[0.40, 0.92)$ and plateau is detected, the system queues a structured HITL review with: dimension breakdown, blocking factors, LLM-generated explanation, and suggested actions (approve finality, provide resolution, escalate, or defer).

4.5 Finality States

State	Trigger	Action
RESOLVED	Score ≥ 0.92 + monotonic gate passed	Automatic completion
ACTIVE	Score < 0.92 or gate not satisfied	Continue processing
ESCALATED	Risk ≥ 0.75 or ≥ 3 contradictions or $\alpha < -0.05$	Human intervention
BLOCKED	≥ 5 idle cycles + ≥ 300s without updates	Stalled
EXPIRED	No activity for 30 days	Process expired
HITL Review	Score $\in [0.40, 0.92)$ + plateau	Structured human decision

Table 2: Finality states and their triggers.

5 Agent Architecture

Six agents operate independently within governance constraints:

Facts Agent. Extracts claims, goals, and risks from unstructured context using a DSPy-based LLM pipeline with optional GLiNER2 named-entity recognition and NLI-based contradiction detection. Outputs typed nodes to the semantic graph with confidence scores and source references.

Drift Agent. Computes changes from baseline across four drift types (contradiction, goal misalignment, factual inaccuracy, entropy). Classifies severity (none/low/medium/high) and routes to governance for policy evaluation.

Planner Agent. Synthesizes facts and drift into ranked proposals. Maps drift dimensions to required next actions, considers pressure-directed routing to bottleneck dimensions, and proposes state transitions with justification.

Status Agent. Tracks $V(t)$, α, monotonicity gate, and plateau detection. Routes near-finality cases to HITL review with full convergence analysis.

Governance Agent. Evaluates all proposals against declarative rules. Routes through approval mode (YOLO/MITL/MASTER). Enforces OpenFGA-based access control. Blocks high-drift transitions; allows low-drift; escalates medium.

Executor. Implements approved actions via atomic state transitions with epoch CAS. Records in append-only Context WAL. Updates semantic graph with monotonic upserts.

Key invariant: Agents do not call each other directly. They emit events that trigger governance rules. This decoupling enables resilience (agent failure does not cascade), auditability (every interaction is logged), and scalability (agents can be independently scaled via hatchery-based dynamic provisioning using M/M/c queueing theory [2]).

6 Comparison with Agent Frameworks

Property	**AutoGen**	**CrewAI**	**LangGraph**	**Ours**
Coordination model	Chat patterns	Hierarchical	State machine	Declarative policy
Shared persistent state	No	No	Partial	Yes (WAL + graph)
Formal convergence	No	No	No	Yes (Lyapunov)
Contradiction tracking	No	No	No	Yes (semantic graph)
Immutable audit log	No	No	No	Yes (append-only)
Declarative governance	No	No	No	Yes (YAML)
Fine-grained access control	No	No	No	Yes (OpenFGA)
CRDT monotonic semantics	No	No	No	Yes
Human-in-the-loop finality	Manual	Manual	Manual	Structured + context
Agent topology	Predefined	Predefined	Predefined	Emergent (policy)

Table 3: Comparison with existing agent coordination frameworks.

The fundamental difference is architectural: existing frameworks *wire agents into topologies* (chains, trees, graphs) where coordination is embedded in code. Our system *governs agents through policy over shared state*, allowing coordination patterns to emerge from rules rather than being predetermined.

This distinction matters for enterprise adoption: when business requirements change (new compliance rules, additional risk factors, different approval workflows), our system requires YAML edits. Existing frameworks require code changes, testing, and redeployment.

7 Validation: Project Horizon

7.1 Scenario: M&A Due Diligence

Project Horizon simulates due diligence on NovaTech AG, a B2B SaaS firm in supply chain compliance. Five documents are ingested sequentially, each introducing new facts and contradictions:

1. **Analyst Briefing:** Baseline facts (EUR 50M ARR, 7 patents, 45% CAGR). Low finality score ($\sim$0.15–0.30).

2. **Financial DD:** ARR revised to EUR 38M—a EUR 12M discrepancy. High contradiction drift triggers governance block, preventing the cycle from advancing silently.

3. **Technical Assessment:** CTO and 2 senior engineers departing. Medium drift escalates to risk committee review.

4. **Market Intelligence:** Patent infringement suit filed. Multiple unresolved contradictions push toward ESCALATED finality state.

5. **Legal Review:** Partial contradiction resolution. Goal score crosses 0.75 (near-finality) but remains below 0.92 (auto-finality). HITL review queued with structured decision options: approve finality, provide resolution, escalate, or defer.

7.2 Key Governance Moments

- **Cycle blocked after Document 2:** Financial discrepancy is not silently absorbed. Governance rules prevent advancement until contradiction is addressed.
- **Investigation recommended:** Contradiction drift signals formal investigation.
- **Risk escalation:** Critical personnel departures routed to structured escalation path.
- **Near-finality HITL:** System has sufficient data to recommend but insufficient confidence for autonomous decision. Human receives dimension breakdown, blockers, and suggested actions.

7.3 Results

Metric	Value
Input documents	5 (analyst, financial, technical, market, legal)
Facts extracted	312
Contradictions identified	18
Agent-resolved contradictions	15 (83%)
Contradictions routed to human	3 (17%)
Total convergence rounds	12
Wall-clock time	240 sec
Immutable audit events	487
Policy rule evaluations	156
Governance blocks triggered	3

Table 4: Project Horizon results.

7.4 Testing and Benchmarks

197 unit tests across 26 files covering: state graph CAS transitions (11), convergence tracker stability (28), finality evaluator logic (11), governance agent modes (18), embedding pipeline (11), and supporting infrastructure.

7 convergence benchmarks validating mathematical properties independently of infrastructure:

Scenario	Rounds	Outcome
Steady convergence ($\sim$5%/round)	15	Monotonic, converging, ETA ≈ 0
Plateau at 0.70 ($\pm$0.002 oscillation)	10	Stagnation detected
Spike-and-drop (0.95 $\rightarrow$ 0.70)	–	Monotonicity gate blocks premature finality
Divergence (contradictions increase)	–	Negative α, zero forward progress
One-dimension bottleneck	–	Pressure identifies blocker dimension
Fast convergence ($\geq$0.92 in 3 rounds)	3	No false plateau during rapid improvement
Empty graph (no claims/goals)	–	Safe defaults, no division by zero

Table 5: Convergence benchmarks validating formal properties.

7.5 Known Validation Gaps

The project explicitly acknowledges: no stress testing for high throughput, no chaos engineering for service failures, single-scope E2E only, synthetic convergence data (not real LLM trajectories), and unmeasured LLM oversight quality.

8 Enterprise Applicability

8.1 Compliance and Auditability

Regulatory frameworks (SOX, GDPR, HIPAA) require demonstrating who made decisions, when, and based on what evidence.

Traditional approaches: Orchestration logic is in code. Auditors must inspect source control, trace execution logs, and reconstruct reasoning manually.

Our approach: Governance rules are human-readable YAML, versioned and approved. All claims and decisions are in the immutable Context WAL. Query the semantic graph: "Who decided NovaTech's ARR was EUR 38M, and what documents justified that?"

8.2 Operational Resilience

Agent failures do not cascade. When one agent is slow or fails, others continue operating. The disagreement metric $V(t)$ rises, triggering escalation rules or human routing. The three-node cycle with epoch CAS ensures atomic transitions even under concurrent agent execution.

8.3 Governance Agility

When business needs change, update the YAML governance rules. The change is auditable (versioned), testable (new rules are evaluated against the existing event log), and live immediately. No code recompilation, no redeployment.

8.4 Dynamic Scaling

The hatchery subsystem uses M/M/c queueing theory with KEDA-style lag monitoring to determine optimal agent instance counts per role. Erlang-style supervisors (one-for-one restart) handle agent failures. This enables elastic scaling without architectural changes.

9 The Reasoning Agent Paradigm

The emergence of reasoning-capable LLM agents—systems that can plan, hypothesize, and revise conclusions based on new evidence—fundamentally changes the coordination problem. In the DAG paradigm, agents are execution units: they receive inputs and produce outputs in a predefined sequence. In the reasoning paradigm, agents are *epistemic actors*: they maintain beliefs, generate hypotheses, and update them as evidence accumulates.

This shift has three implications for coordination:

Contradiction is information, not failure. When two reasoning agents disagree about a fact (e.g., conflicting revenue figures from different documents), this disagreement is valuable signal. A governance-based system can track contradictions explicitly in the semantic graph, route them to investigation, and declare finality only when they are resolved—or escalated to human judgment with full context.

Coordination must be emergent, not hardcoded. Reasoning agents discover new information that may invalidate the planned execution path. A governance-based system allows agents to propose new directions based on what they find, evaluated by declarative rules rather than predetermined sequences. The three-node cycle (ContextIngested $\rightarrow$ FactsExtracted $\rightarrow$ DriftChecked) provides structure without rigidity.

Finality requires formal guarantees. When agents reason and revise, the system may never converge without formal mechanisms. Lyapunov-based convergence tracking provides mathematical guarantees that the system either reaches consensus or explicitly routes to human review—it cannot cycle indefinitely.

Our system is designed for this paradigm: agents reason independently over shared context, governance rules coordinate their outputs, and formal mechanisms ensure convergence. As LLM agents become more capable (chain-of-thought reasoning, tool use, multi-step planning), the governance-based coordination model becomes more relevant, not less.

10 Limitations and Future Work

Scalability: Validation at 50 claims with 30% logical overlap. Production scale (thousands of claims, hundreds of agents) requires semantic graph optimization (approximate nearest-neighbor search, event log sharding, compiled governance rules).

Heterogeneous agents: Current agents are LLM-based. Extending to symbolic reasoners, neural networks, and human experts requires standardized agent interfaces and cross-modality convergence criteria.

Real LLM trajectories: Convergence benchmarks use synthetic data. Validation with real LLM reasoning traces (with their inherent stochasticity) remains future work.

Byzantine agents: Current convergence proofs assume agents make progress. Handling adversarial or faulty agents requires Byzantine-tolerant consensus machinery.

Multi-scope coordination: Current validation is single-scope. Enterprise deployments will require cross-scope governance and convergence.

11 Conclusion

We have presented a paradigm for autonomous agent coordination based on declarative governance over shared immutable context. By replacing rigid pipelines with YAML-defined policy rules, maintaining CRDT-inspired monotonic state, and providing Lyapunov-based convergence guarantees, we achieve a system that is simultaneously auditable, resilient, explainable, and scalable.

Unlike existing agent frameworks that wire agents into predefined topologies, our system lets coordination patterns emerge from policy. Agents reason independently over shared context, propose transitions evaluated by declarative rules, and converge through formal mechanisms—or route to structured human review when autonomous consensus is insufficient.

As LLM agents become increasingly capable of reasoning, planning, and hypothesis generation, the need for governance-based coordination—rather than pipeline-based orchestration—will only grow. We believe this work establishes foundations for enterprise-grade multi-agent systems where auditability, compliance, and formal convergence are not afterthoughts but architectural primitives.

Acknowledgments

This work was supported by Deal ex Machina SAS. We thank the communities behind PostgreSQL, NATS, OpenFGA, DSPy, and Mastra for foundational infrastructure.

References

[1] Alberto Camacho et al. MACI: EMA-based stagnation detection in multi-agent coordination. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

[2] Marco Dorigo et al. Swarm intelligence: From stigmergic coordination to pressure-directed activation. *Swarm Intelligence*, 18(1):1–28, 2024.

[3] Haibin Duan et al. Aegean: Monotonicity gates and coordination invariants for multi-agent convergence. 2025. Preprint.

[4] Shadaj Laddad et al. CodeCRDT: Monotonic merge operations for distributed collaborative editing. In *Proceedings of the ACM on Programming Languages (OOPSLA)*, 2024.

[5] LangChain Team. LangGraph: Building stateful, multi-actor applications with LLMs. `https://github.com/langchain-ai/langgraph`, 2024.

[6] João Moura. CrewAI: Framework for orchestrating role-playing autonomous AI agents. `https://github.com/joaomdmoura/crewAI`, 2024.

[7] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[8] Ruoming Pang et al. Zanzibar: Google's consistent, global authorization system. In *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC)*, pages 33–46, 2019.

[9] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.