# Deterministic Control Over Stochastic Agents in Tick-Based Social Simulation

Marcos Pazzarelli
socialcompute.dev

February 27, 2026

**Status:** Working paper draft
**Scope note:** This document is implementation-grounded and intentionally avoids proprietary details (prompts, thresholds, and internal enforcement recipes).

**Abstract**

We present a social simulation architecture where LLMs act strictly as stochastic proposal engines while a deterministic core remains the only authority over state transitions. The key design is a hard generation/authorization boundary: proposals are validated, repaired, degraded, or rejected before any canonical state update. The system runs in discrete ticks with ordered phases (perception, proposal, validation, resolution, post-update), explicit agent state vectors (beliefs, memory with provenance, social state, reply obligations), and engine-governed reflection that cannot overwrite world truth.

Determinism is defined operationally at multiple levels. In the strongest practically useful form, the simulator is deterministic given identical initial state and an identical proposal trace (including recorded model outputs), producing identical trajectories and audit logs. We argue that this pattern improves epistemic integrity, long-horizon coherence, and observability, and we outline an evaluation agenda based on ablations and failure-mode metrics. The paper documents the architecture and experimental protocol design without disclosing proprietary prompts, thresholds, or internal enforcement recipes.

## 1 Introduction

LLM-based agents are effective at producing locally plausible behavior, but they are unreliable as direct authorities over simulation state. In multi-agent social environments, this mismatch produces recurring failure modes: unsupported assertions, epistemic leakage, conversational loops, state drift, and brittle long-horizon behavior. These failures are not merely stylistic. They compromise the validity of causal trajectories and weaken the interpretability of simulation outcomes.

We present an alternative design in which the language model is treated as a stochastic proposal mechanism operating under deterministic control. The simulation engine owns canonical state, enforces world constraints, and governs all executable transitions. This separation supports two goals simultaneously: preserving behavioral variability at the surface while maintaining hard guarantees on feasibility, timing, and epistemic consistency.

The contribution of this paper is architectural. We describe a practical control pattern for social simulation that combines (1) explicit state vectors, (2) deterministic validation and action mediation, (3) forced epistemic propagation through state updates rather than prompt residue, and (4) telemetry/replay instrumentation for debugging and auditability.

# 2 System Model

## 2.1 Tick-Based Simulation Core

The simulator advances in discrete ticks. Each tick executes a structured pipeline with ordered phases, including:

- time advancement,
- exogenous world updates,
- per-agent perception construction,
- agent decision proposal,
- deterministic action validation,
- action resolution,
- post-resolution social and emotional updates,
- persistence/replay capture.

This phase discipline is central. It prevents out-of-order state mutations and makes it possible to reason about causality, replay trajectories, and intervention points.

## 2.2 Determinism Levels

To avoid ambiguity, we define determinism operationally at two levels:

- `D0 (Trace-Deterministic Replay)`: Given the same initial state, the same RNG state, and the same proposal trace (i.e., model outputs are recorded/replayed), the simulator produces identical state trajectories and audit logs.
- `D1 (Provider-Dependent Sampling Determinism)`: Given the same model identifier, sampling parameters, and call order, the simulator is expected to reproduce the same proposal trace. In practice, this may fail if provider-side behavior changes.

The architectural claim of this paper relies on `D0`, not on stronger assumptions about provider-side invariance.

## 2.3 Separation of Generation and Authorization

Agent actions originate as proposals, which may be generated by an LLM-backed policy or a deterministic baseline policy. However, proposals are not executable by default. Each proposal is passed through a deterministic validator that checks world feasibility, co-presence, epistemic support, conversational obligations, and social-physics constraints. The validator can:

- accept a proposal as valid,
- normalize it (e.g., canonicalizing targets),
- repair it (e.g., reframe unsupported assertions into lower-commitment forms),
- degrade it into a non-state-changing action,

- reject it entirely.

This implements a strict control boundary: the model may propose intent, but only the engine can authorize state change.

### 2.4 Agent State Vectors and Canonical Memory

Agents are modeled with structured state vectors rather than relying on prompt-only memory. The state representation includes:

- beliefs (including suspect distributions and known facts),
- memory items with provenance and confidence,
- relationship metrics,
- emotional state,
- pending reply obligations and short-lived conversational queues,
- strategy/reflection state for private planning.

This design supports durable state across long runs and reduces dependence on reconstructing context from compressed prompt history. Knowledge enters the system through explicit events, observations, inferences, and validated communication, not merely because a model generated a sentence.

## 3 Epistemic Control and Agency Sequestration

### 3.1 Conceptual Control-Layer Interface

The control layer can be described with a small, implementation-agnostic interface:

```
Proposal {
  actor_id,
  type,
  target_ref?,
  content?,
  intent?,
  evidence_refs[],
  meta{}
}

ArbitrationDecision {
  outcome: ACCEPT | REPAIR | DEGRADE | REJECT,
  reason_codes[],
  patched_proposal?,
  annotations{}
}
```

Minimal arbitration pseudocode:

```
for each tick:
  perceptions <- build_perceptions(world, agents, recent_events)
  proposals <- policy.generate(perceptions)
  for p in proposals:
    d <- arbitrate(p, world, agent_state, social_constraints, epistemic_constraints)
    executable <- apply_decision(p, d)
  resolve(executable_actions)
  update_state_vectors_and_logs()
```

This abstraction is enough to communicate engineering structure without disclosing thresholds, prompts, or proprietary repair heuristics.

### 3.2 Perception Scoping as an Epistemic Boundary

Perception is built from local visibility and recent relevant events. Agents only receive information that is compatible with their spatial position, local co-presence, and event exposure. Hidden objects remain hidden unless discovered; remote actors are not visible; local events can be perceived and incorporated into memory. This mechanism functions as an operational epistemology: what an agent can say or infer is constrained by what the simulation has made available to that agent.

### 3.3 Deterministic Validation as Agency Sequestration

The validator is the main instrument for agency sequestration. It constrains action execution by checking:

- physical feasibility (movement paths, locks, adjacency, occupancy conditions),
- social feasibility (co-presence, audience requirements, interaction saturation),
- epistemic basis (supporting evidence, known facts, memory traces),
- conversational coherence (reply obligations, anti-loop rules, anti-echo guards),
- mode-specific constraints (e.g., staged dramatic states such as climax/meeting logic).

This means an agent can be behaviorally creative while still being unable to perform impossible, unsupported, or coherence-destroying actions.

### 3.4 Repair Instead of Only Rejection

A notable property of the control layer is that it does not rely on binary accept/reject behavior alone. Unsupported or over-assertive utterances can be repaired into safer forms (for example, transformed into questions, hypotheses, or lower-commitment speech acts). This preserves interaction flow while preventing canon inflation through unsupported claims. This is especially important in social simulation, where over-aggressive blocking can collapse dialogue variety, but permissive acceptance can quickly destroy epistemic integrity.

### 3.5 Forced Conversational Closure and Intra-Tick Coherence

The system includes explicit mechanisms to preserve same-tick conversational coherence. If one agent asks a question and another attempts to deflect by moving away or redirecting to a third party, the engine can prioritize the conversational obligation, interrupt movement, or force reply

priority. This prevents a common failure mode in unconstrained agents: open-thread accumulation without closure. The result is not scripted conversation, but bounded conversational dynamics in which obligations become part of the executable state.

### 3.6 Trace Example: Epistemic Repair and Target Override

The following excerpt documents intervention by the control layer on a proposal that violates epistemic constraints. In tick 2, an agent (Aurelius) attempts to assert a temporal fact about the victim ("I then walked to the kitchen to grab a glass of water, and that's when I found her unresponsive") without any supporting evidence references in its state vector. Rather than rejecting the action and freezing the agent, the deterministic arbiter rewrites the claim and degrades the speech act into a hypothesis.

**Stochastic proposal (LLM raw_response).**


```
{
  "action": "talk",
  "target": "Rodrigo",
  "speech_act": "ANSWER",
  "utterance_public": "I was in the study, going over some papers, when I last saw Lucia.
      I then walked to the kitchen to grab a glass of water, and that's when I found her
      unresponsive.",
  "evidence_refs": []
}
```


**Canonical executed resolution (engine events_json).**


```
{
  "type": "talk",
  "tick_index": 2,
  "room_id": "living_room",
  "agent_id": "88886839_aurelius",
  "target_id": "88886839_rodrigo",
  "data": {
    "utterance": "It's unclear what Lucia was doing in the kitchen before she became
        unresponsive.",
    "auto_repair": "TALK_CLAIM_UNSUPPORTED",
    "repair_kind": "TALK_CLAIM_UNSUPPORTED",
    "rewrite_mode": "hypothesis",
    "speech_act_primary": "HYPOTHESIS"
  }
}
```


Similarly, the engine enforces physical rigor: movement proposals that attempt to traverse topologically invalid edges (e.g., trying to pass into a locked room) are intercepted with `MOVE_BLOCKED_DOOR_LOCKED` and silently degraded into a `wait` event, preserving the spatial integrity of the world without breaking the simulation loop.

## 4 Reflection Under Deterministic Governance

The architecture supports scheduled reflective planning passes ("System 2" style reflection), but reflection is governed by the engine and runs as a designated simulation phase rather than free recursion. Reflection outputs are normalized to a schema and incorporated into private strategy state under deterministic post-processing. This design has two consequences:

- reflection can improve local planning and uncertainty management without blocking the main simulation loop,
- reflection cannot directly overwrite canonical world truth or bypass normal action validation.

In other words, reflection is a controlled planning mechanism, not a privileged state mutation channel.

# 5 Observability, Replay, and Debuggability

## 5.1 Telemetry for LLM Calls and Tick Outcomes

The system persists tick snapshots and agent snapshots, along with structured telemetry for LLM calls (including call purpose, provider/model identifiers, timing, retry metadata, and policy metadata). This creates an audit surface that is useful for debugging model behavior in context rather than treating LLM inference as an opaque side effect.

## 5.2 Replayable Simulation State

The architecture supports replay and state reconstruction from persisted snapshots. This is operationally important in long-running simulations because it enables:

- post hoc inspection of failure trajectories,
- reproducible debugging from checkpoints,
- comparison of policy changes over equivalent starting conditions.

## 5.3 Controlled Exposure and Redaction

The system distinguishes between public, omniscient, and debug-oriented views of simulation state and telemetry. Private agent memory and raw prompt/response artifacts can be redacted in non-debug views. This separation allows the same simulation infrastructure to support both observability and information hygiene.

# 6 Reliability Controls for LLM-Backed Policies

The control stack includes explicit guardrails around LLM calls, including tick-level budgeting, timeout handling, and fallback behaviors. The practical goal is not to eliminate model failures, but to ensure that transient failures (timeouts, retries, backoff events, or budget exhaustion) are absorbed by the deterministic simulation loop rather than causing uncontrolled collapse of agent behavior.

A key design choice is graceful fallback: if a model call is blocked by guardrails, the engine can still produce a deterministic action fallback instead of reducing the agent to permanent inactivity. This preserves simulation continuity while keeping control authority in the engine.

# 7 Engineering Workflow

This architecture implies a different engineering workflow than prompt-only agent design. The primary work shifts toward:

- defining state representations,
- writing and refining validators and repair paths,
- calibrating social and conversational pressure systems,
- designing observability surfaces and replay tools,
- iterating on failure cases using telemetry and audits.

The relevant unit of progress is not only "better prompts," but more reliable mediation between generation and executable simulation law.

In practice, the workflow is driven by three recurring loops:

- `Replay-driven debugging`: inspect tick replays and state transitions to localize where coherence breaks.
- `Failure taxonomy refinement`: classify recurrent failures (unsupported assertions, leakage, looping, open-thread accumulation) and map them to control-layer interventions.
- `Repair-operator iteration`: add or improve repair/degrade operators that preserve flow while protecting canonical state.

# 8 Toy Evaluation Protocol (Minimal, Reproducible)

To move from architectural description to empirical evidence without requiring a large benchmark, we propose a minimal controlled evaluation:

## 8.1 Fixed Scenario

Use a single fixed social mystery scene with:

- a closed environment,
- multiple agents,
- hidden information,
- evidence discovery opportunities,
- conversational and accusatory interactions.

The purpose is not broad coverage, but stress-testing epistemic integrity and conversational control under a stable setting.

## 8.2 Baselines

Compare at least two execution modes:

- `Baseline A (Unmediated/Weakly Mediated)`: policy proposals are resolved with minimal epistemic and conversational validation.
- `Baseline B (Governed)`: full control layer enabled (validation, repair/degrade paths, reply-priority enforcement, reflection governance).

Optional ablations can disable individual mechanisms (e.g., repair operators or same-tick reply enforcement) while keeping others fixed.

## 8.3 Metrics (Failure-Mode Oriented)

Report simple, interpretable metrics per run:

- `Unsupported Assertion Rate`: fraction of utterances/actions asserting facts without valid epistemic basis.
- `Epistemic Leakage Rate`: fraction of agent outputs referencing information not available within that agent's accessible state.
- `Open-Thread Accumulation`: number of unresolved question/response obligations over time.
- `Loop Rate`: frequency of repeated or near-repeated conversational patterns without new evidence/state gain.

These metrics are aligned with the architecture's claims (integrity, coherence, auditability) rather than generic task-completion scores.

### 8.3.1 Operational Metric Definitions (Log-Derivable)

To keep the evaluation reproducible under `DO`, the following metrics are defined in terms of engine logs, state vectors, and arbitration traces (not subjective human ratings). Implementations may vary, but each metric should be computable from persisted tick snapshots and action traces.

- **Unsupported Assertion Rate.** Fraction of talk/actions that commit to a world fact without support in the agent's accessible state at that tick. Operationally: count outputs whose referenced entities/claims are not present in (i) the agent's perception scope, (ii) agent memory items with matching provenance, or (iii) explicit `evidence_refs` attached to the proposal; normalize by total talk/actions.
- **Epistemic Leakage Rate.** Fraction of outputs that reference information outside the agent's epistemic boundary. Operationally: for each output, extract referenced entities/events (preferably via structured fields; otherwise via deterministic entity-linking on content) and test membership in the tick's `accessible_entity_set` for that agent (derived from perception + validated memory). Leakage is any reference not in the set.
- **Open-Thread Accumulation.** Unresolved conversational obligations over time. Operationally: track the size of `pending_replies` (or equivalent obligation queue) per agent per tick and report (i) mean, (ii) max, and optionally (iii) area-under-curve (AUC) across the run.

- **Loop Rate.** Repetition of conversational patterns without new state gain. Operationally: detect sequences of near-duplicate utterances/actions (content similarity above a fixed threshold) where the associated `state_delta` introduces no new evidence_refs, no new discoveries, and no new canonical events beyond repeated talk. Report loops per 100 ticks (or per N actions).

These definitions intentionally avoid proprietary thresholds: the only requirement is that thresholds and extractors are fixed per experimental setup and reported alongside results.

## 8.4 Reporting Standard

A minimal paper-ready report can include:

- one fixed scene definition,
- 2 baselines,
- 4 metrics,
- short qualitative trace excerpts (redacted if needed),
- ablation table with aggregate rates over multiple seeded runs.

## 8.5 Qualitative Results and Autopsy

### 8.5.1 Autopsy of Inaction in Unmediated Policies

The complete failure to resolve the scenario in the unmediated model (`BASELINE_A_RAW`) is not due to physical inactivity. The event log shows a very high volume of movement proposals (715 `move` events) and dialogue (462 `talk` events). The failure lies in the absence of closing transitions in the social automaton. Unlike the governed engine, which recorded multiple critical deterministic events (`accuse`=4, `confess`=4, `call_meeting`=7), the raw configuration executed none of these actions. The LLM can generate perpetual exploration and dialogue, but it is structurally unable to converge to a terminal state transition without pressure from an external deterministic core.

### 8.5.2 Failure Absorption and Engine Liveness

The robustness of the architecture is demonstrated by its handling of silent stochastic failures. In approximately 10% of calls, the provider returned empty or null text without emitting an HTTP error code. In a naive LLM-agent design, this would corrupt state or halt time progression. In the governed engine trace, these outputs are absorbed by injecting deterministic preservation events (e.g., `type: wait` with `degradation_reason: CLIMAX_THREAD_LOCK_WAIT`). The agent state vector evolves correctly through the corresponding delta and the tick advances, guaranteeing system liveness under silent foundation-model failures.

### 8.5.3 Sustained Conversational Debt

Time-series analysis of the `reply_queue` state refutes the hypothesis of an exponential explosion of open threads, but reveals an equally destructive phenomenon: persistent conversational debt. In the ablation without reply priority (`ABLATION_NO_REPLY_PRIORITY`), the pending-obligation queue does not decay. In the final window of the simulation (ticks 41–51), the governed architecture forces closure and achieves a mean of 0.173 pending obligations, while the ablation retains a late-stage queue of 0.275. This residual load fragments agent attention and prevents consolidation of the shared evidence needed to resolve the scenario.

## 9 Related Work (Positioning)

This work sits at the intersection of LLM agents, multi-agent orchestration, and simulation infrastructure.

- `Generative Agents`-style systems foreground memory and reflection, but typically do not impose a deterministic authorization boundary over every state transition in a simulated world.
- Multi-agent orchestration frameworks (e.g., tool- and conversation-oriented agent frameworks such as AutoGen [2]) provide coordination patterns, but are not by themselves a social-physics engine with canonical state authority.
- Agent benchmarks (e.g., AgentBench-style evaluations [3]) motivate reproducible metrics and infrastructure, but often emphasize task completion over epistemic integrity in long-horizon social trajectories.
- Large-scale social simulation efforts with many agents [4] illustrate the field's direction and scale ambitions, reinforcing the need for control architectures and audit surfaces as systems grow.

The distinguishing claim here is not merely memory or orchestration, but deterministic mediation between stochastic proposals and executable world updates.

## 10 Limitations

This paper is an architectural report, not a benchmark study. It does not provide comparative metrics against external agent frameworks, nor does it claim universal transfer across domains. Several limitations remain:

- control-layer complexity grows with domain richness,
- over-constraint can reduce behavioral diversity,
- calibration remains scenario-dependent,
- reflective planning can add runtime overhead even when bounded.

These are engineering trade-offs, not contradictions of the approach.

## 11 Future Evaluation Agenda

A full empirical version of this work should include:

- ablation studies for validation, repair, and conversational closure mechanisms,
- comparisons between unconstrained and governed policy execution,
- long-horizon coherence and epistemic-consistency metrics,
- stress tests under model latency and transient failure conditions,
- replay-based error taxonomy and remediation loops.

The core claim to test is not that deterministic control makes agents "smarter," but that it makes complex social simulations more coherent, more inspectable, and more reliable as infrastructure.

## 12 Conclusion

The central lesson of this system is architectural: LLMs are useful proposal generators, but social simulation requires a deterministic authority over state transitions. By separating generation from authorization, maintaining canonical state vectors, and enforcing epistemic constraints through executable validation, it becomes possible to retain behavioral variability without surrendering causal coherence. This suggests a broader direction for social computing: progress will depend not only on larger models, but on stronger control architectures that govern how model outputs enter and transform simulated worlds.

## A Architecture Pipeline

The following diagram illustrates the tick-based execution pipeline, isolating the stochastic LLM calls from the canonical state mutations governed by the deterministic validator.

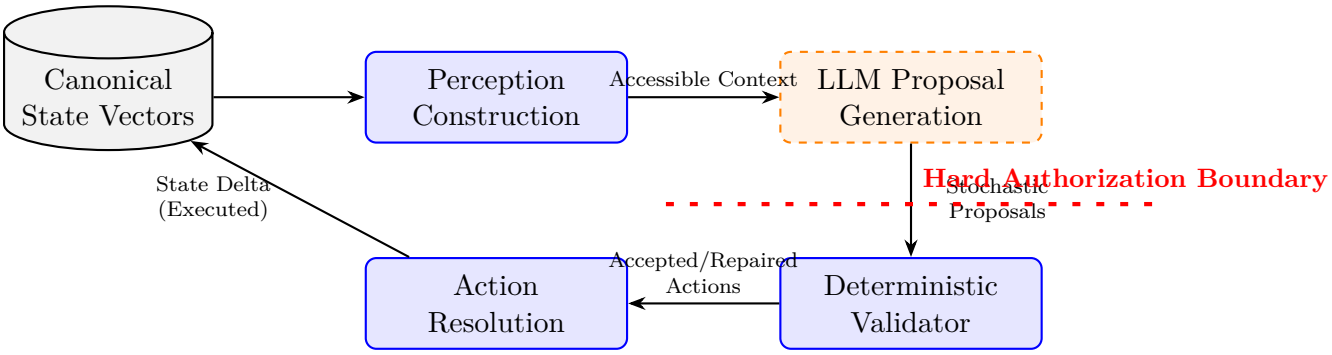


Figure 1: Tick-based simulation pipeline separating stochastic generation from deterministic state updates.

## References


[1] 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:2304.03442, 2023.

[2] 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 Framework.* arXiv:2308.08155, 2023.

[3] Xian Liu, Wenhao Yu, Han Zhang, et al. *AgentBench: Evaluating LLMs as Agents.* arXiv:2308.03688, 2023.

[4] Joon Sung Park, Carolyn Q. Zou, Aaron Shaw, Benjamin Mako Hill, Carrie J. Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S. Bernstein. *Generative Agent Simulations of 1,000 People.* arXiv:2411.10109, 2024.