Spherepop OS: A Deterministic Semantic Operating System Formal Specification and Architectural Rationale

Flyxion

December 13, 2025

Abstract

Spherepop OS is a semantic operating system whose primary abstraction is not the process, file, or thread, but a deterministic, append-only relational event substrate. The system is designed to support collaborative, time-aware, and introspectable computation by enforcing strict causal ordering, replayability, and separation between authoritative state transitions and derived views. This document presents a formal specification of Spherepop OS, together with the rationale for each architectural choice. Particular emphasis is placed on determinism, observational purity, and the treatment of geometry, layout, and speculation as non-causal metadata layered atop an immutable event log. The resulting system occupies an intermediate position between an operating system kernel, a distributed database, and a semantic field theory of computation.

1 Motivation and Design Goals

Contemporary operating systems are historically organized around mutable global state, imperative control flow, and opaque side effects. While these designs have proven effective for single-user, single-machine computation, they exhibit deep structural limitations when extended to collaborative, distributed, or semantically rich environments. In particular, traditional systems struggle to provide strong guarantees of determinism, introspectability, and time-consistent reasoning.

Spherepop OS is motivated by the following design goals:

G1. **Deterministic Replay**: Any system state must be reproducible exactly from an authoritative log.

G2. **Total Causal Order**: All state transitions must admit a global, unambiguous ordering.

G3. **View–Cause Separation**: Observations, visualizations, and speculative reasoning must not interfere with authoritative state.

G4. **Semantic Primacy**: Relations, equivalence, and meaning precede processes and files as first-class concepts.

G5. **Collaborative Introspection**: Multiple observers may join, leave, rewind, and speculate without destabilizing the system.

These goals jointly rule out large classes of conventional kernel and database architectures, motivating a log-centric, event-sourced design with strict discipline around causality and derivation.

2 Core Ontology

Definition 2.1 (Semantic Object). *A semantic object is an abstract entity identified by a stable handle. Objects have no intrinsic meaning except insofar as they participate in relations, equivalence classes, and events.*

Definition 2.2 (Relation). *A relation is a typed, directed association between two semantic objects. Relations are first-class and may carry flags or metadata.*

Definition 2.3 (Equivalence). *An equivalence relation over objects identifies multiple handles as representing a single canonical representative. Equivalence is induced by merge events and maintained via a union–find structure.*

Importantly, objects, relations, and equivalence classes exist only as consequences of event replay. There is no mutable object table independent of the log.

3 Authoritative Event Log

Axiom 3.1 (Append-Only Authority). *All authoritative state transitions in Spherepop OS are recorded as events in a single append-only log. No other structure is permitted to introduce or modify semantic state.*

Each event is assigned a strictly increasing sequence identifier (EID) by a single authoritative arbiter. The log is therefore totally ordered.

Proposition 3.1 (Deterministic State). *Given an initial empty state and a prefix of the event log, the resulting semantic state is uniquely determined.*

Proof. All kernel transitions are pure functions of prior state and the next event. Since the log is totally ordered and immutable, replay is deterministic. □

This property elevates the log from an implementation detail to the primary artifact of the system.

4 Kernel Semantics

The Spherepop kernel is a deterministic interpreter of events. It maintains derived state such as equivalence classes and relation tables, but these structures are caches, not authorities.

Definition 4.1 (Kernel). *The kernel is the minimal deterministic machine that replays events and emits derived changes (diffs). It has no external side effects and no hidden state.*

Kernel operations include:

- Object creation (POP)
- Merge (MERGE, inducing equivalence)

  - Relation creation (LINK) and removal (UNLINK)
  - Region collapse (COLLAPSE)

- Metadata attachment (SET$_M ETA$)
  Each operation is represented as a canonical event type with a fixed binary schema.

5 Formal Execution Semantics

We now make the kernel transition function explicit by presenting a small-step operational semantics for event application. The goal is not to specify implementation details, but to provide an unambiguous mathematical description of how authoritative state evolves in response to events.

5.1 Kernel State

Definition 5.1 (Kernel State)**.** *A kernel state is a tuple*
$\sigma = (O, U, R, M)$

- *O is a finite set of object identifiers,*
- *$U : O \to O$ is a union–find parent map representing equivalence,*
- *$R \subseteq O \times O \times \mathcal{T}$ is a multiset of typed relations,*
- *$M : O \to \mathcal{K} \rightharpoonup \mathcal{V}$ is a partial metadata map.*

We write $\mathrm{rep}_\sigma(o)$ for the canonical representative of o under U, after path compression.

5.2 Transition Relation

Kernel execution is defined by a labeled transition relation
$\sigma \xrightarrow{e} \sigma'$

5.3 POP (Object Creation)

$$\mathrm{o} \notin O \frac{}{(O,U,R,M) \xrightarrow{\mathrm{POP}(o)} (O \cup \{o\},\, U[o \mapsto o],\, R,\, M)}$$

5.4 MERGE (Equivalence Induction)

$$\mathrm{o}_1, o_2 \in O \frac{}{\sigma \xrightarrow{\mathrm{MERGE}(o_1,o_2)} \sigma'}$$
where σ' differs from σ only in U, with
U'(rep(o$_2$)) = rep(o_1).
This rule is idempotent: merging already-equivalent objects yields no further change.

5.5 LINK (Relation Creation)

$$(\mathrm{o}_1, o_2, t) \notin R \frac{}{\sigma \xrightarrow{\mathrm{LINK}(o_1,o_2,t)} (O,U,R \cup \{(\mathrm{rep}(o_1),\mathrm{rep}(o_2),t)\},M)}$$

5.6 UNLINK (Relation Removal)

$$(o_1, o_2, t) \in R \overline{\sigma \xrightarrow{\text{UNLINK}(o_1,o_2,t)} (O,U,R \setminus \{(\text{rep}(o_1),\text{rep}(o_2),t)\},M)}$$

5.7 COLLAPSE (Bulk Equivalence)

Let $S \subseteq O$ be a region to be collapsed, with chosen representative $o_r \in S$.

$$S \subseteq O \overline{\sigma \xrightarrow{\text{COLLAPSE}(S,o_r)} \sigma'}$$

where for all $o \in S$, $U'(\text{rep}(o)) = \text{rep}(o_r)$. Objects not equal to o_r remain in O but are no longer representatives.

5.8 SET$_M ETA(MetadataAttachment)$

$$o \in O \overline{\sigma \xrightarrow{\text{SET\_META}(o,k,v)} (O,U,R,M[o \mapsto (k \mapsto v)])}$$

Metadata updates do not affect O, U, or R.

5.9 Semantic Properties

Proposition 5.1 (Determinism). *For any kernel state σ and event e, there exists at most one state σ' such that $\sigma \xrightarrow{e} \sigma'$.*

Proposition 5.2 (Merge Confluence). *The final equivalence relation induced by a set of MERGE events is independent of their order.*

Proof. Union–find union is associative and commutative over equivalence classes. The induced partition is given by the reflexive–transitive closure of the merge relation and therefore does not depend on merge order. □

Proposition 5.3 (View Consistency). *Derived views that collapse non-representative objects preserve semantic equivalence.*

5.10 MERGE-Induced Relation Normalization

We now make explicit the interaction between equivalence induction and relations.

Axiom 5.1 (Representative Normalization). *All relations are stored and interpreted in representative-normalized form. That is, for any relation $(o_1, o_2, t) \in R$, it is invariant that $o_1 = \text{rep}(o_1)$ and $o_2 = \text{rep}(o_2)$.*

Proposition 5.4 (Relation Rewriting under MERGE). *When an event* MERGE(o_a, o_b) *is applied, all relations incident to* $\text{rep}(o_b)$ *are rewritten to reference* $\text{rep}(o_a)$.

Proof. After MERGE, $\text{rep}(o_b) = \text{rep}(o_a)$. Any relation previously referencing $\text{rep}(o_b)$ is therefore normalized to reference $\text{rep}(o_a)$. No semantic information is lost, since o_a and o_b are equivalent by definition. □

6 Replay Equivalence

We now formalize the equivalence between full replay and incremental observation.

Theorem 6.1 (Replay Equivalence)**.** *Let $\ell = e_1, e_2, \ldots, e_n$ be an event sequence. Let σ_{replay} be the state obtained by replaying ℓ from the initial state σ_0. Let σ_{diff} be the state obtained by applying the sequence of diffs derived from $e_1, \ldots, e_n$ to σ_0. Then σ_{replay} and σ_{diff} are semantically equivalent.*

Proof. Each diff is a deterministic function of a single event application. Since kernel transitions are deterministic and diffs are derived without side effects, incremental application of diffs reconstructs the same representative-normalized state as direct replay of the log. Equivalence follows by induction on the length of ℓ. □

Proposition 6.1 (Merge Confluence)**.** *The final equivalence relation induced by a set of MERGE events is independent of their order.*

Proposition 6.2 (View Consistency)**.** *Derived views that collapse non-representative objects preserve semantic equivalence.*

7 Meta-Theorem: Functoriality of Derived Views

The core philosophical discipline of Spherepop OS is the separation of *cause* (events) from *views* (snapshots, diffs, geometry). The following meta-theorem states this separation as a functoriality principle: any admissible view is a structure-preserving map out of the event-sourced semantics.

7.1 Event Prefix Category

Definition 7.1 (Prefix Category)**.** *Fix an event log $\ell = (e_1, e_2, \ldots)$. Define the* prefix category **Pref**(ℓ) *as follows.*

- *Objects are natural numbers $n \in \mathbb{N}$, interpreted as prefixes $\ell_{\leq n} := (e_1, \ldots, e_n)$.*
- *There is a unique morphism $m_{n \to n+k} : n \to n + k$ for each $k \geq 0$, representing prefix extension by k events.*
- *Composition is given by addition: $m_{n \to n+k} \circ m_{n+k \to n+k+j} = m_{n \to n+k+j}$.*

7.2 State Semantics as a Functor

Definition 7.2 (State Semantics)**.** *Let* **State** *be the category whose objects are kernel states σ and whose morphisms are deterministic state updates induced by event application. Define the* state semantics functor

$$\mathsf{S}_\ell : \mathbf{Pref}(\ell) \to \mathbf{State}$$

by $\mathsf{S}_\ell(n) := \sigma_n$, the state obtained by replaying $\ell_{\leq n}$ from σ_0, and by mapping each generator $m_{n \to n+1}$ to the single-step transition

$$\sigma_n \xrightarrow{e_{n+1}} \sigma_{n+1}.$$

Proposition 7.1 (Functoriality of Replay)**.** *The assignment* S_ℓ *is a functor.*

Proof. Identity morphisms correspond to applying zero events and therefore map to identity state updates. Composition in $\mathbf{Pref}(\ell)$ corresponds to sequential event application, which equals composition of deterministic transitions in **State** by definition. □

7.3 Derived Views

Definition 7.3 (Admissible View)**.** *An* admissible view *is any functor*

$$\mathsf{V} : \mathbf{State} \to \mathbf{View}$$

into some category of observer representations **View** *(e.g. JSON graphs, NDJSON diffs, geometric layouts), such that* V *is* non-interfering*: it does not feed back into* S_ℓ*.*

Theorem 7.1 (View Functoriality Meta-Theorem)**.** *For any log* ℓ *and any admissible view functor* $\mathsf{V} : \mathbf{State} \to \mathbf{View}$*, the composite*

$$\mathsf{V} \circ \mathsf{S}_\ell : \mathbf{Pref}(\ell) \to \mathbf{View}$$

exists and is a functor. In particular:

(i) ***Causal Respect:*** *view updates compose according to event order;*

(ii) ***Snapshot Coherence:*** $\mathsf{V}(\sigma_n)$ *is uniquely determined by the prefix* $\ell_{\leq n}$*;*

(iii) ***Transport Independence:*** *any two admissible transports realizing the same* V *are observationally equivalent;*

(iv) ***Gauge Freedom:*** *if* V *quotients by a metadata gauge (e.g. layout hints), semantic content is preserved.*

Proof. Since S_ℓ is a functor and V is a functor, their composite is a functor. Items (i)–(iv) are immediate restatements: functoriality enforces respect for composition (event order), object mapping enforces prefix-determined snapshots, transport choices refine the same arrow in **View**, and gauge quotienting corresponds to functoring into a category that identifies gauge-equivalent metadata. □

Corollary 7.1 (Diffs and Snapshots as Two Views)**.** *Let* V_{snap} *be the view that serializes full state (snapshot) and* V_{diff} *be the view that serializes incremental changes. Both are admissible views, hence both factor functorially through* S_ℓ*. The Replay Equivalence theorem is the statement that these two views agree on semantic content up to the observer's equivalence relation.*

Corollary 7.2 (Late-Joiner Correctness)**.** *A late-joining observer that first receives* $\mathsf{V}_{\text{snap}}(\sigma_k)$ *and then receives* V_{diff} *for subsequent events reconstructs the same view as an observer that applied* V_{diff} *from the beginning.*

8 Diffs and Incremental Observation

While the log is authoritative, clients require efficient incremental observation.

Definition 8.1 (Diff)**.** *A diff is a derived, non-authoritative description of how kernel state changes as a result of applying a single event.*

Diffs may include object additions, removals, relation changes, or metadata updates. They are broadcast to observers but are never logged themselves.

Axiom 8.1 (View Non-Interference)**.** *Diffs do not influence kernel state and may be dropped, reordered, or ignored by observers without affecting correctness.*

This separation permits high-frequency visualization without compromising determinism.

9 Snapshots as Derived Views

Definition 9.1 (Snapshot)**.** *A snapshot is a complete serialization of kernel state derived by replaying a prefix of the event log.*

Snapshots are used for bootstrapping late-joining clients and for historical inspection.

Proposition 9.1 (Snapshot Purity)**.** *Snapshots introduce no new information beyond what is contained in the log prefix they represent.*

Snapshots are never logged and never assigned sequence identifiers. They are observational artifacts only.

10 Seekable Time and Historical Inspection

Spherepop OS supports time navigation by allowing clients to request snapshots at arbitrary past EIDs.

Definition 10.1 (Seek-to-EID Snapshot)**.** *A seekable snapshot is a snapshot derived by replaying the log up to a specified EID.*

This operation is implemented using a temporary kernel instance, ensuring isolation from live state.

Corollary 10.1. *Historical inspection cannot affect the present state of the system.*

11 Speculative Branches

Speculation is treated as a local, non-authoritative overlay.

Definition 11.1 (Speculative Branch)**.** *A speculative branch consists of a base EID and a client-local overlay log of hypothetical events.*

Branches are replayed by applying the authoritative log up to the base EID, followed by the overlay events. Branches may be discarded or rebased freely.

Axiom 11.1 (Speculative Isolation)**.** *Speculative events have no effect on authoritative state unless explicitly re-submitted as proposals.*

This design permits safe exploration without timeline pollution.

12 Layout and Geometry as Metadata

Spherepop OS distinguishes sharply between semantic structure and geometric presentation.

Definition 12.1 (Layout Hint)**.** *A layout hint is non-causal metadata attached to an object, expressing advisory geometric information such as position, scale, or clustering intent.*

Layout hints may be modified freely by clients and are never interpreted by the kernel as semantic operations.

Proposition 12.1. *Geometry in Spherepop OS is a gauge choice, not a semantic constraint.*

This permits rich visualization while preserving semantic invariants.

13 Arbiter and Collaboration Model

The arbiter is responsible for assigning sequence identifiers and appending events to the log.

Definition 13.1 (Arbiter)**.** *The arbiter is the sole authority permitted to commit events to the log.*

Clients submit proposals, which are either accepted and sequenced or rejected. Accepted proposals are broadcast as diffs to all observers, including the originator.

Axiom 13.1 (Single Sequencer)**.** *There exists exactly one arbiter per authoritative log.*

This ensures total causal order.

14 Rationale and Comparative Analysis

Spherepop OS differs from traditional operating systems and databases in several respects:

- Unlike POSIX kernels, all state is replayable and inspectable.
- Unlike CRDT-based systems, a single total order is enforced.
- Unlike version control systems, speculation is first-class and continuous.

The architecture may be understood as an operating system for semantic time rather than for hardware resources.

15 Conclusion

Spherepop OS presents a disciplined rethinking of operating system structure grounded in determinism, causal clarity, and semantic primacy. By elevating the event log to the status of sole authority and rigorously separating causes from views, the system supports collaboration, introspection, and speculative reasoning without sacrificing correctness. The design intentionally leaves room for future extensions, including richer semantic types, entropy-driven scheduling, and distributed arbitration, while preserving its core invariants.