

# From Sets to Parts: Operational Mereology via Event–Sourced Semantics

Flyxion

December 13, 2025

## Abstract

Set theory has long served as the default foundational language of mathematics and computation, encoding structure through element–membership relations. This essay argues that the Spherepop calculus and kernel provide a viable alternative foundation based on mereology: a theory of part–whole relations grounded in operational, replayable semantics. By replacing extensional membership with event–induced composition, Spherepop eliminates several pathologies of set theory while retaining expressive power sufficient for computation, logic, and structure.

## 1 The Foundational Role of Sets

Set theory traditionally supplies:

- a notion of collection (via membership),
- identity (via extensionality),
- structure (via subsets and relations),
- and abstraction (via comprehension).

However, these roles are not neutral. Membership is static, global, and atemporal; extensional equality identifies objects by total content rather than by construction or history. For both computation and ontology, this introduces difficulties: paradoxes of unrestricted comprehension, non-constructive existence, and a mismatch between formal identity and operational behavior.

In practice, modern computing systems rarely manipulate sets in this sense. They manipulate *states*, *records*, *logs*, and *relations*, all of which are temporal and operational.

## 2 Mereology as an Alternative

Mereology replaces membership ( $\in$ ) with a primitive *part-of* relation ( $\leq$ ). Instead of asking whether  $x$  is an element of  $S$ , we ask whether  $x$  is a part of  $y$ .

Classical mereology has long been proposed as an alternative foundation, but it often remains abstract and axiomatic, lacking a concrete operational realization. Spherepop supplies exactly this missing component.

### 3 Spherepop’s Ontological Commitments

Spherepop commits to the following primitives:

1. **Objects** identified by stable handles.
2. **Relations** linking objects (e.g. links, merges).
3. **Events** as the sole source of authoritative change.
4. **Replay** as the mechanism of state reconstruction.

Crucially, there is no primitive notion of a set. There is no global membership relation. All structure arises from the accumulation of events and the relations they induce.

### 4 Part–Whole as Event Semantics

In Spherepop, part–whole relations are not axiomatic; they are *constructed*.

- A POP event introduces a new part.
- A MERGE event establishes that one object subsumes another.
- A LINK event introduces structured adjacency without subsumption.
- A COLLAPSE event explicitly equates or eliminates parts.

Thus, “being part of” is not a predicate evaluated against a set, but a historical fact witnessed by replay.

### 5 Replacing Membership

Set-theoretic membership

$$x \in S$$

is replaced by the operational statement:

$$x \leq y \quad \text{iff replay shows that } x \text{ was merged into or linked under } y.$$

This has several consequences:

- There is no unrestricted comprehension.
- No object has parts it did not acquire through events.
- Identity is historical, not extensional.

In particular, two objects may be extensionally identical at a moment in time but remain distinct if their event histories differ.

## 6 Temporalized Identity

Set theory identifies objects by total membership. Spherepop identifies objects by:

1. their handle,
2. their event history,
3. their induced relations at a given replay point.

Identity is therefore *time-indexed*. This resolves classical puzzles of extensional collapse and aligns identity with computation.

## 7 Authority and Mereological Disagreement

Spherepop’s arbiter introduces an additional dimension absent from set theory: *authority*.

Different authorities may propose conflicting part–whole relations. The arbiter orders, accepts, or rejects these proposals, yielding a single committed mereology. Disagreement is not inconsistency; it is a first-class phenomenon.

This replaces the absolute, god’s-eye perspective of classical set theory with an explicitly mediated ontology.

## 8 Why Sets Become Unnecessary

Once we have:

- objects,
- relations,
- part–whole operations,
- replayable history,

the traditional roles of sets are subsumed:

- collections become aggregates of parts,
- subsets become substructures,
- membership becomes ancestry or containment,
- abstraction becomes projection or view.

No loss of expressive power occurs; instead, the ontology becomes constructive and inspectable.

## 9 Relation to Logic and Mathematics

Spherepop does not deny classical mathematics; rather, it reinterprets it.

Sets may still be used as *views* or *summaries* of a state, but they are no longer foundational. They are derived artifacts, not ontological commitments.

This mirrors how relational databases replaced set-theoretic tables with transactional logs, without loss of query power.

## 10 Conclusion

Spherepop demonstrates that a replayable, event-sourced mereology can replace set theory as a foundation for computation and structure. By grounding part-whole relations in operational history rather than extensional membership, it resolves long-standing mismatches between mathematical abstraction and computational reality.

Sets are not abolished; they are demoted from foundations to conveniences.

## 11 A Formal Mereological Axiom System Induced by Spherepop

This section extracts an explicit axiom system whose models correspond exactly to replayable Spherepop states. Unlike classical mereology, the axioms are not postulated abstractly but are induced by the operational constraints of the Spherepop kernel.

### 11.1 Motivation

Classical mereology replaces set membership with a part-whole relation, but it typically remains axiomatic and timeless. Spherepop enforces a *constructive* and *temporal* mereology: parts exist only through witnessed events, and part-whole relations are indexed by replay time.

The goal of this section is to formalize this induced mereology without introducing ontological commitments beyond those already enforced by the kernel.

### 11.2 Primitive Symbols

We assume the following primitives:

- A countable set of object identifiers (handles), written  $x, y, z$ .
- A totally ordered set of event indices (or ticks), written  $t \in \mathbb{T}$ .
- A replay operator  $\mathcal{R}(t)$  mapping an event prefix to a state.
- A binary relation  $\leq_t$  (“ $I$  is part of at time  $t$ ”).

No membership relation  $\in$  is assumed. All structure arises from replay.

### 11.3 Axiom M1: Existence by Construction

**Statement.** An object  $x$  exists at time  $t$  if and only if there exists an event  $e \leq t$  that introduces  $x$ .

**Formalization.**

$$\exists_t(x) \iff \exists e \leq t \text{ such that } e = \text{POP}(x)$$

**Interpretation.** There are no primitive or implicit objects. Existence is not a logical quantifier but a historical fact.

### 11.4 Axiom M2: Temporal Part-Of

**Statement.** The part-of relation is time-indexed and induced by replay.

**Formalization.**

$$x \leq_t y \iff \mathcal{R}(t) \text{ witnesses a merge or containment path from } x \text{ to } y$$

**Interpretation.** Part-of is not primitive; it is computed from event history. Different times may induce different mereologies.

### 11.5 Axiom M3: No Implicit Parts

**Statement.** An object may acquire parts only through explicit events.

**Formalization.**

$$x \leq_t y \Rightarrow \exists e \leq t \text{ such that } e \text{ contributes to } x \leq y$$

**Interpretation.** There is no comprehension principle. No object contains parts merely because they satisfy a predicate.

### 11.6 Axiom M4: Weak Antisymmetry

**Statement.** Mutual containment does not imply identity unless explicitly asserted.

**Formalization.**

$$(x \leq_t y \wedge y \leq_t x) \Rightarrow \exists e \leq t \text{ such that } e = \text{COLLAPSE}(x, y)$$

**Interpretation.** Extensional equality is replaced by explicit identification. Cycles are permitted but never implicit.

## 11.7 Axiom M5: Replay Determinism

**Statement.** Replay is deterministic.

**Formalization.** If two event sequences share the same prefix up to  $t$ , then:

$$\mathcal{R}_1(t) = \mathcal{R}_2(t)$$

**Interpretation.** The induced mereology is a function of history alone, not of interpretation or observation.

## 11.8 Derived Properties

From the axioms above, several properties follow.

**Conditional Transitivity.** If  $x \leq_t y$  and  $y \leq_t z$ , then  $x \leq_t z$ , provided no intervening event breaks the containment path.

**Well-Foundedness in Time.** No object may be part of another object at a time prior to its own introduction.

**Explicit Cycles.** Cycles in the part-of graph may exist, but only if created by explicit events.

## 11.9 Model Theorem

**Theorem.** Every finite Spherpap event log induces a unique temporal mereological structure satisfying Axioms M1–M5.

**Proof Sketch.** By construction: replay defines existence (M1), induces part-of relations (M2), forbids uncaused relations (M3), preserves identity distinctions unless collapsed (M4), and is deterministic (M5). No additional structure is required.

## 11.10 Comparison with Classical Mereology

Classical mereology typically assumes:

- timeless part-of,
- unrestricted supplementation,
- extensional identity.

Spherpap rejects all three. Time, construction, and authority replace abstract closure conditions.

### 11.11 Summary

Spherepop induces a constructive, temporal mereology in which:

- objects exist only by construction,
- parts are acquired historically,
- identity is explicit,
- and replay replaces axiomatic closure.

This mereology is not an interpretation layered atop the system; it is the system.

## 12 Relation to Martin–Löf Type Theory

This section compares the operational mereology induced by Spherepop with Martin–Löf Type Theory (MLTT). The aim is not to subsume one framework into the other, but to clarify how Spherepop replaces several roles played by types, identity types, and universes with event–sourced structure.

### 12.1 Types as Sets vs Objects as Handles

In MLTT, a type is often interpreted as a collection of terms, and typing judgments  $a : A$  are read as membership statements. Even in intensional settings, types function as domains of admissible inhabitants.

Spherepop rejects this interpretation. There are no domains of inhabitants. Instead:

- objects are introduced by explicit events,
- handles provide stable identity,
- admissibility is determined by replayed state, not by typing.

Thus, where MLTT uses types to delimit existence, Spherepop uses construction.

### 12.2 Identity Types and Explicit Collapse

MLTT internalizes equality via identity types  $Id_A(x, y)$ , whose inhabitants are proofs witnessing equality. These proofs are first-class and may be nontrivial even when  $x$  and  $y$  are definitionally equal.

Spherepop replaces identity types with explicit **COLLAPSE** events.

#### Correspondence.

- An inhabitant of  $Id_A(x, y)$  corresponds to a witnessed identification.
- A **COLLAPSE**( $x, y$ ) event performs this identification at the ontological level.

**Key Difference.** In MLTT, identity is a logical object. In Spherepop, identity is an irreversible historical act unless further events intervene.

### 12.3 Dependent Types vs Metadata

Dependent types  $B(x)$  allow the structure of a type to depend on a term. This provides expressive power but also introduces complex proof obligations.

Spherepop instead uses metadata attached to objects or relations:

- metadata fields may depend on existing structure,
- metadata does not introduce new objects by itself,
- metadata is non-authoritative unless acted upon by events.

Thus, where dependent types internalize dependency into the type system, Spherepop externalizes dependency into time-indexed annotations.

### 12.4 Induction and Replay

MLTT supports induction as a fundamental principle: objects are constructed by constructors, and proofs proceed by induction over these constructors.

Spherepop replaces induction with replay.

**Observation.** Replay unfolds a structure step by step, mirroring the constructive reading of inductive definitions.

**Difference.** Induction reasons over all possible constructions; replay reasons over actual constructions. The former is logical, the latter historical.

### 12.5 Universes and Authority Layers

MLTT introduces universes to stratify types and avoid paradox. These universes are themselves types, requiring careful handling.

Spherepop replaces universes with authority layers:

- multiple authorities may propose structures,
- the arbiter orders and commits events,
- higher-level structure arises from governance, not stratified typing.

Authority replaces universes as the mechanism that prevents uncontrolled self-reference.



## 12.6 Judgmental Equality vs Replay Equivalence

MLTT distinguishes judgmental (definitional) equality from propositional equality.

Spherepop instead distinguishes:

- byte-level equality of events,
- replay equivalence of prefixes,
- explicit collapse equivalence of objects.

There is no implicit equality. All equivalence is either syntactic or event-induced.

## 12.7 Expressive Power and Tradeoffs

**Where MLTT is Stronger.**

- internal proof theory,
- higher-order abstraction,
- rich dependent constructions.

**Where Spherepop is Stronger.**

- operational semantics,
- inspectable identity,
- temporal structure,
- authority mediation.

## 12.8 Summary

Spherepop and MLTT address overlapping foundational concerns using different strategies. MLTT internalizes structure into logic; Spherepop externalizes structure into history. Types become views over replayed state, not ontological containers. Identity becomes an event, not a proof.

This shift trades logical generality for operational clarity, a trade that is appropriate for systems grounded in computation rather than proof.

## 13 Why Russell-Style Paradoxes Cannot Be Formulated

This section shows that paradoxes of the Russell type cannot even be stated within the Spherepop framework. This is not because they are resolved by additional axioms, but because the ontological and syntactic prerequisites required to formulate them are absent by construction.

### 13.1 The Structure of Russell’s Paradox

Russell’s paradox relies on three ingredients:

1. A universal domain of objects.
2. A primitive membership relation  $\in$ .
3. An unrestricted comprehension principle allowing objects to be defined by predicates.

Formally, the paradox constructs the set

$$R = \{x \mid x \notin x\}$$

and derives a contradiction by asking whether  $R \in R$ .

### 13.2 Absence of Membership

Spherepop has no primitive membership relation. There is no syntactic form corresponding to  $x \in y$ . The closest analogue, the part-of relation  $\leq_t$ , is:

- time-indexed,
- induced by replay,
- applicable only to existing objects.

As a result, the expression  $x \leq_t x$  is neither globally meaningful nor paradoxical; it is merely a question about historical containment, answerable only by replay.

### 13.3 No Universal Quantification Over Objects

Russell’s construction quantifies over *all* sets. Spherepop provides no such domain.

At any time  $t$ , only finitely many objects exist, and they exist only as a consequence of events. There is no closed universe of discourse over which predicates may range.

Thus, there is no syntactic space in which a predicate like “does not contain itself” could range over all objects.

### 13.4 No Predicate-Generated Objects

Most importantly, Spherepop forbids predicate-generated existence.

**Constraint.** Objects may only be introduced by POP events.

**Consequence.** No object can be defined as “the object satisfying property  $\varphi$ ”. Properties may be evaluated, but they do not generate handles.

Russell’s paradox fails at the construction step: there is no operation that turns a predicate into an object.

### 13.5 No Self-Reference Without Handles

Self-reference in Russell’s paradox depends on an object being able to refer to itself through membership.

In Spherepop:

- handles are assigned externally,
- an object cannot name itself without an explicit event,
- circularity must be deliberately introduced.

Thus, even self-containment is not paradoxical; it is merely an explicit structure, visible and auditable.

### 13.6 Theorem: Russell Objects Are Inexpressible

**Theorem.** There exists no Spherepop expression or event sequence that corresponds to the Russell set

$$\{x \mid x \not\prec_t x\}.$$

**Proof Sketch.** The expression presupposes (i) a membership predicate, (ii) quantification over all objects, and (iii) predicate-based object construction. None of these are available in Spherepop. Therefore the construction cannot be expressed.

### 13.7 Generalization to Impredicative Paradoxes

The same reasoning applies to other impredicative constructions:

- Burali–Forti paradox,
- Cantor’s diagonal argument (as ontology),
- unrestricted power sets.

All rely on turning totality into an object. Spherepop strictly separates totality (as a view) from existence (as an event).

### 13.8 Summary

Spherepop does not *solve* Russell’s paradox. It prevents its formulation. By eliminating unrestricted comprehension, global membership, and predicate-generated objects, paradoxes based on self-reference are structurally impossible.

Consistency is achieved not by axiomatic restriction, but by ontological discipline.

## 14 Computational Complexity: Power Sets versus Event Logs

This section argues that the replacement of set-theoretic foundations with event-sourced mereology is not merely philosophically motivated, but computationally necessary. In particular, it shows that axioms such as Power Set encode worst-case complexity into ontology itself, whereas Spherepop’s event-log foundation scales linearly with construction.

### 14.1 The Computational Cost of Power Sets

The power set axiom asserts that for any set  $X$ , there exists a set  $\mathcal{P}(X)$  containing all subsets of  $X$ .

From a computational perspective, this entails:

- exponential growth:  $|\mathcal{P}(X)| = 2^{|X|}$ ,
- implicit materialization of all possible substructures,
- non-local dependency: adding an element to  $X$  changes  $\mathcal{P}(X)$  globally.

When interpreted ontologically rather than as a convenience, the power set axiom commits the system to exponential structure regardless of whether it is ever observed or used.

### 14.2 Event Logs as Linear Structures

Spherepop replaces power sets with append-only event logs.

Let  $E_t$  denote the prefix of the event log up to time  $t$ . Then:

- $|E_t|$  grows linearly with the number of events,
- state is reconstructed by replaying  $E_t$ ,
- no implicit objects exist beyond those introduced by events.

Complexity is therefore proportional to actual construction, not to hypothetical possibility.

### 14.3 Queries versus Reification

In set-theoretic foundations, subsets are reified as first-class objects. In Spherepop, analogous notions are handled by queries over replayed state.

**Example.** The collection of all parts of an object  $x$  is not an object, but the result of a traversal of the part-of graph induced by  $\mathcal{R}(t)$ .

**Consequence.** Computational cost is incurred only when information is requested, not when it is merely possible.

## 14.4 Replay Complexity

Let  $n = |E_t|$  be the number of events up to time  $t$ .

- Replay complexity is  $O(n)$  in the absence of indexing.
- Incremental replay and caching can reduce this to amortized sublinear cost.
- No operation requires enumerating all substructures of an object unless explicitly requested.

This contrasts sharply with power-set semantics, where exponential structure is implicit even when unused.

## 14.5 Case Studies from Practice

The superiority of event-log foundations is already visible in real systems:

- **Databases:** write-ahead logs replace materialized views.
- **Version control:** commits replace snapshots of all possible histories.
- **CRDTs:** merge operations replace global state enumeration.

Spherepop unifies these practices into a foundational ontology rather than treating them as engineering compromises.

## 14.6 Power Sets as Ontological Overcommitment

The power set axiom encodes a maximalist view of existence: everything that could exist does exist.

Spherepop adopts the opposite stance:

Only what has been constructed exists; everything else is a computation.

This shift moves complexity from ontology to algorithms, where it can be controlled, optimized, and reasoned about.

## 14.7 Theorem: Linear Ontology, Polynomial Computation

**Theorem.** Let  $E_t$  be a Spherepop event log of length  $n$ . Then the ontological size of the system is  $O(n)$ , and all derived structures are computable in time polynomial in  $n$ .

**Proof Sketch.** Only events contribute to existence. All derived structures are computed by traversal, replay, or projection over  $E_t$ . No axiom introduces super-polynomially many entities.

## 14.8 Summary

Set-theoretic foundations bake worst-case complexity into existence itself. Spherepop defers complexity to computation, ensuring that the cost of structure is proportional to actual construction. This is not merely an optimization; it is a foundational correction aligned with how computation truly operates.

## 15 Category Theory as a View Layer over Event–Sourced Mereology

This final section situates Spherepop with respect to category-theoretic foundations. Rather than proposing category theory as an alternative foundation, Spherepop treats categorical structure as a *derived view* over replayable event logs. Categories arise from observation, not from ontology.

### 15.1 Why Category Theory Appears Foundational

Category theory is often proposed as a foundation because it:

- abstracts away from element-level detail,
- replaces membership with morphisms,
- emphasizes compositional structure.

These motivations align closely with Spherepop’s rejection of set membership and embrace of relational structure. However, category theory typically presupposes:

- a fixed collection of objects,
- timeless morphisms,
- global composition laws.

Spherepop provides these only as *derived* notions.

### 15.2 Objects as Event-Induced Nodes

Given a replay time  $t$ , define a graph  $G_t$  whose nodes are object handles existing at  $t$ , and whose edges are relations induced by replay.

From this graph, one may define a category  $\mathcal{C}_t$ :

- objects of  $\mathcal{C}_t$  are handles in  $G_t$ ,
- morphisms are paths generated by links and merges,
- identity morphisms are trivial paths.

This category exists only relative to  $t$ ; it is not an eternal structure.

### 15.3 Morphisms as Relations, Not Primitives

In Spherepop, relations are induced by events. Morphisms are therefore:

- historical,
- contingent,
- revocable by later events.

Composition is path concatenation in the replayed graph, not a primitive operation. Associativity holds only insofar as replay produces a stable graph.

Thus, categorical laws are *emergent invariants*, not axioms.

### 15.4 Colimits as Explicit Merges

Many categorical constructions correspond directly to Spherepop operations.

**Colimits.** A merge operation corresponds to a colimit only when explicitly introduced by a MERGE event.

**Key Distinction.** Category theory guarantees the existence of colimits by axiom; Spherepop requires their construction.

This eliminates non-constructive existence while preserving expressive power.

### 15.5 Limits as Queries

Limits, pullbacks, and products are not objects in Spherepop unless explicitly constructed. Instead, they are queries over replayed state.

For example, a pullback corresponds to identifying objects related to two others by traversal, not to the creation of a new universal object.

This mirrors database joins rather than ontological commitments.

### 15.6 Functors as Projections

Views in Spherepop naturally correspond to functors:

- from the replay-induced category  $\mathcal{C}_t$ ,
- to a category of representations (tables, graphs, metrics).

These functors are:

- time-indexed,
- non-authoritative,
- discardable without ontological loss.

Thus, category theory becomes a language of observation rather than construction.

## 15.7 Natural Transformations as Metadata Evolution

Metadata updates induce coherent changes across views. These correspond to natural transformations between functors indexed by time.

However, such transformations:

- do not introduce new objects,
- do not alter the underlying mereology,
- are secondary to event authority.

This demotes naturality from an axiom to a property of well-behaved projections.

## 15.8 Why Spherepop Is Pre-Categorical

Spherepop precedes category theory in the following sense:

- it defines existence before arrows,
- it defines construction before composition,
- it defines authority before universality.

Category theory may be layered atop Spherepop, but cannot replace its event-sourced ontology.

## 15.9 Summary

Category theory remains an indispensable descriptive language. Spherepop clarifies its proper role: not as a foundation of existence, but as a powerful, optional lens through which replayed structure may be analyzed.

Foundations describe what exists. Categories describe how we look.

# 16 Historical Background: From Parts and Wholes to Operational Mereology

The philosophical investigation of parts and wholes long predates its formalization as mereology. Questions concerning composition, persistence, and structure appear at every stage of the history of philosophy, often arising precisely where set-like or collection-based thinking proves inadequate.

## 16.1 Ancient and Medieval Precursors

In ancient philosophy, mereological questions were inseparable from metaphysics itself. The Pre-Socratic schools disagreed fundamentally about whether reality admits of parts at all. The Eleatics denied genuine plurality, the Atomists grounded all structure in indivisible units, and the Pluralists attempted to reconcile unity and multiplicity through compositional principles.



Aristotle introduced a distinction that would remain central: a whole is not merely the sum of its parts, but possesses a unity that cannot be reduced to aggregation alone. This insight underlies classical puzzles such as the Ship of Theseus, which asks how an object can persist through the gradual replacement of its parts. The puzzle is not about identity in the abstract, but about the relation between historical change and structural continuity.

Medieval philosophers developed these issues further. Figures such as Peter Abelard, John Buridan, and Thomas Aquinas debated whether wholes are identical to their parts or something over and above them. Abelard, unusually, defended a strict summative view according to which the whole just is the sum of its parts, combined with a form of mereological essentialism: every part is essential to the whole to which it belongs. Although historically marginal, this position highlights a tension that persists into modern foundations—whether structure is intrinsic or imposed.

## 16.2 The Emergence of Mereology as a Formal Discipline

The term *mereology* (from the Greek *meros*, meaning part or share) was introduced in the early twentieth century by the Polish logician Stanisław Leśniewski. Leśniewski developed mereology as part of a broader foundational program intended to replace set theory with a nominalistically acceptable alternative. His aim was not merely technical elegance, but ontological restraint: to avoid commitment to abstract entities such as sets or classes.

Leśniewski's work was later independently reinvented and popularized by Henry S. Leonard and Nelson Goodman. Goodman, in particular, viewed mereology as a way to reconstruct mathematics and ontology without the Platonic commitments of set theory. In this sense, mereology emerged not as a metaphysical curiosity, but as a foundational rival to set-theoretic membership.

In parallel, Edmund Husserl developed a theory of parts and wholes within his project of *formal ontology*. In the *Logical Investigations* (1900–1901), Husserl sought a subject-matter-independent theory of objects as such. For Husserl, mereology was not an alternative to logic, but a prerequisite for it: any account of objects must explain how parts compose wholes across material, event-like, and abstract domains.

These two strands—Leśniewski's nominalism and Husserl's formal ontology—established mereology as a serious foundational framework rather than a regional metaphysical theory.

## 16.3 Mereology and the Limits of Set-Theoretic Foundations

Despite its promise, classical mereology remained largely axiomatic and timeless. Part-whole relations were treated as static facts, abstracted away from processes of construction, change, or authority. As a result, mereology often functioned as a conceptual alternative to set theory without providing a concrete operational replacement.

Set theory, by contrast, became entrenched not because of ontological plausibility, but because of its convenience as a uniform representational language. Membership, power sets, and comprehension offered expressive completeness at the cost of non-constructive existence and paradox-management by axiom restriction.

From the perspective of computation, this tradeoff has always been suspect. Real systems do not manipulate power sets; they manipulate histories, logs, transactions, and relations that evolve

over time.

## 16.4 Spherepop in Historical Context

Spherepop may be understood as the operational realization of the nominalist and formal-ontological ambitions that motivated early mereologists. Like Leśniewski and Goodman, it rejects set-theoretic membership as foundational. Like Husserl, it treats mereology as domain-general. Unlike both, it grounds part-whole structure in replayable event history.

In Spherepop:

- parts are acquired through events rather than axioms,
- wholes persist through historical continuity rather than extensional identity,
- and structure is governed by explicit authority rather than implicit closure.

This places Spherepop within a long philosophical tradition while also marking a decisive shift: mereology becomes executable. The theory of parts and wholes is no longer merely a metaphysical alternative to set theory, but a computational substrate capable of replacing it.

## 16.5 Summary

The history of mereology reveals a recurring dissatisfaction with membership-based foundations and an enduring concern with composition, persistence, and structure. Spherepop inherits these concerns and resolves them by introducing time, construction, and authority as first-class features. In doing so, it transforms mereology from a static axiom system into a dynamic foundation aligned with contemporary computational reality.

## 17 Conclusion

This paper has argued that the foundational role traditionally played by set theory can be assumed instead by an operational mereology grounded in event-sourced semantics. Rather than treating collections, membership, and abstraction as primitive, the Spherepop framework replaces them with construction, replay, and part-whole relations that are explicit, temporal, and auditable.

The central shift is not merely formal but ontological. In Spherepop, existence is historical rather than axiomatic; identity is induced by events rather than extensional content; and structure is something that happens, not something that is postulated. Set-theoretic notions reappear only as views—useful projections over replayed state—rather than as commitments about what fundamentally exists.

By examining the relationship between Spherepop and classical mereology, type theory, set theory, and category theory, the paper has shown that many of the strengths attributed to set-theoretic foundations are preserved while several of their most problematic features are eliminated. Unrestricted comprehension, power-set ontology, and impredicative self-reference are not repaired by additional axioms; they are rendered inexpressible by design. Consistency is achieved through constructional discipline rather than logical restriction.

The computational argument reinforces this conclusion. Set-theoretic foundations encode worst-case complexity into ontology, committing systems to structures whose size and cost bear no relation to actual use. Event logs, by contrast, grow in proportion to what is done rather than what is possible. Spherepop therefore aligns foundational commitments with the realities of computation, where history, replay, and authority are unavoidable.

Importantly, Spherepop does not deny the value of existing mathematical frameworks. Type theory, category theory, and set theory remain powerful descriptive and analytic tools. What changes is their status: they become languages for reasoning about replayed structure, not the source of that structure. Foundations are relocated from abstract universes to executable processes.

Several open questions remain. The precise expressiveness of event-sourced mereology relative to higher-order logics warrants further study, as does the relationship between authority-mediated construction and proof-theoretic notions of choice and independence. These questions, however, are now framed within a system whose primitives are inspectable and whose commitments are explicit.

In replacing sets with parts, Spherepop does not offer a new metaphysical picture of the world so much as a new discipline of construction. What exists is what has been built. What is known is what can be replayed. And what is abstract is what is derived, not assumed.

## A From ZF Axioms to Operational Mereology

This appendix provides an explicit correspondence between the axioms of Zermelo–Fraenkel set theory (ZF) and the operational mereology implemented by the Spherepop kernel and calculus. The goal is not to refute ZF, but to show that each axiom either (i) becomes unnecessary, (ii) is realized operationally, or (iii) is demoted to a derived view.

### A.1 Primitive Ontologies

**ZF.** The primitive ontology consists of sets and the membership relation  $\in$ .

**Spherepop.** The primitive ontology consists of:

- object handles,
- event records,
- a replay function mapping event prefixes to states,
- induced relations between objects.

There is no primitive membership relation. All structure is event-induced.

### A.2 Axiom of Extensionality

**ZF Statement.**

$$\forall x \forall y (\forall z (z \in x \leftrightarrow z \in y) \rightarrow x = y)$$

**Spherepop Correspondence.** There is no global extensional equality. Two objects are equal only if:

1. they share the same handle, or
2. an explicit COLLAPSE event equates them.

**Consequence.** Objects with identical induced parts may remain distinct if their event histories differ. Identity is historical, not extensional.

### A.3 Axiom of Empty Set

**ZF Statement.**

$$\exists x \forall y (y \notin x)$$

**Spherepop Correspondence.** There is no primitive empty object. Every object exists only by virtue of a POP event. The absence of parts is represented by the absence of events, not by a distinguished empty entity.

**Consequence.** Non-existence is represented by non-construction, not by a universal empty container.

### A.4 Axiom of Pairing

**ZF Statement.**

$$\forall a \forall b \exists x (a \in x \wedge b \in x)$$

**Spherepop Correspondence.** Given objects  $a$  and  $b$ , a new object  $x$  may be introduced by POP, followed by MERGE( $a, x$ ) and MERGE( $b, x$ ).

**Consequence.** Pairing is constructive and explicit. No object implicitly contains others without a witnessed event.

### A.5 Axiom of Union

**ZF Statement.**

$$\forall x \exists u \forall y (y \in u \leftrightarrow \exists z (y \in z \wedge z \in x))$$

**Spherepop Correspondence.** Union corresponds to an explicit aggregation object whose parts are acquired via merge events from constituent aggregates. There is no automatic flattening of containment.

**Consequence.** Union is a process, not an axiom. Nested structure remains unless explicitly collapsed.

## A.6 Axiom of Power Set

**ZF Statement.**

$$\forall x \exists p \forall y (y \in p \leftrightarrow y \subseteq x)$$

**Spherepop Correspondence.** There is no power object containing all substructures. Substructures are not reified unless constructed.

**Consequence.** This axiom is intentionally absent. Spherepop forbids the automatic reification of all possible parts, preventing combinatorial explosion and non-constructive existence.

## A.7 Axiom of Separation

**ZF Statement.**

$$\forall x \exists y \forall z (z \in y \leftrightarrow z \in x \wedge \varphi(z))$$

**Spherepop Correspondence.** Separation is realized as a *view*, not an object. Filters over replayed state produce projections without introducing new ontological entities.

**Consequence.** Predicates do not generate objects. They generate queries.

## A.8 Axiom of Replacement

**ZF Statement.** Functional images of sets are sets.

**Spherepop Correspondence.** Transformations over objects produce new objects only when explicitly materialized by events. Otherwise, they remain views or derived computations.

**Consequence.** Replacement becomes a controlled construction principle rather than an implicit closure property.

## A.9 Axiom of Foundation

**ZF Statement.** Every nonempty set has an  $\in$ -minimal element.

**Spherepop Correspondence.** Event order provides a natural well-foundedness: no object may depend on events that occur after its creation.

**Consequence.** Cycles in containment are permitted only when explicitly introduced (e.g. via links), and never arise implicitly.

## A.10 Axiom of Choice

**ZF Statement.** There exists a choice function selecting elements from sets.

**Spherepop Correspondence.** Choice is realized as authority-mediated arbitration. Selection among competing proposals is explicit, weighted, and auditable.

**Consequence.** Choice is operational and contextual, not axiomatic.

## A.11 Summary Table

<b>ZF Axiom</b>	<b>Spherepop Status</b>
Extensionality	Replaced by historical identity
Empty Set	Eliminated
Pairing	Constructive via events
Union	Explicit aggregation
Power Set	Rejected
Separation	View-level only
Replacement	Event-mediated
Foundation	Enforced by time
Choice	Arbiter-mediated

## A.12 Conclusion

Spherepop does not weaken mathematical expressivity; it refactors it. By replacing set-theoretic axioms with operational constraints, it grounds structure in construction, time, and authority. Mereology ceases to be an abstract alternative and becomes an executable foundation.