History as Identity: The Append-Only Algebra of Spherepop

Flyxion

March 6, 2026

Abstract

Many computational models identify a process with its observable input-output behavior and discard the internal sequence of operations that produced that behavior. In systems governed by irreversible commitments, however, this internal sequence often carries essential information about causality and provenance.

This paper introduces *Spherepop*, a minimal algebraic framework in which the identity of a computation is its append-only history of commitments rather than its final state. A Spherepop state is represented by a residual option space together with an accumulated commitment history. Three primitive operators—Pop, Bind, and Collapse—govern the evolution of this structure: Pop records irreversible commitments, Bind restricts admissible options without committing, and Collapse reconstructs observable state from the resulting history.

We show that these operators generate a free history category whose morphisms are ordered commitment sequences, prove that the operator set is minimal, and establish that conventional state semantics arises as a quotient of this structure obtained by identifying histories with identical observable outcomes. This perspective clarifies the structure of systems whose natural representation is append-only, including event-sourced architectures, version control systems, and causal process models.

Contents

1 Introduction

A recurring ambition in the theory of computation is the identification of the minimal mathematical structure required to model a class of phenomena faithfully. For deterministic sequential computation that structure is the Turing machine or equivalently the λ-calculus; for concurrent interaction it is process algebra or the π-calculus; for feedback and recursion within monoidal categories it is the traced monoidal category of Joyal, Street, and Verity. In each case the theory achieves its power by deciding what to abstract over and what to preserve.

Traces, in particular, achieve their power precisely by abstraction. The trace operator $\mathrm{Tr}^{U}_{A,B}(f)$ for a morphism $f : A \otimes U \to B \otimes U$ asserts that the precise internal behavior on the "feedback wire" U is irrelevant once its effect on the external interface $A \to B$ is determined. The loop is collapsed; what remains is the fixed-point input-output relation.

Spherepop originates from the refusal of precisely this identification. Its central commitment is that the internal path—the ordered sequence of decisions, exclusions, and contextual restrictions that produced an observable state—is not auxiliary information to be discarded once a result is known. It is the result. Two computations that terminate at the same value but follow different causal chains are, in Spherepop's ontology, distinct objects. The identity of a process is its history, not its output.

1.1 Contributions

The aim of this paper is to formalize the intuition that, in many systems governed by irreversible commitments, the identity of a process is determined by the ordered history of those commitments rather than by its final state alone. The principal contributions are threefold.

First, we introduce a minimal state representation consisting of a residual option space and an append-only commitment history, capturing the distinction between possibilities that remain available and decisions that have already been made.

Second, we define three primitive operators—Pop, Bind, and Collapse—prove that they constitute a minimal generating set for the history algebra, and establish their principal algebraic properties including monotonicity, non-commutativity, and functoriality of collapse.

Third, we characterize the resulting structure categorically as a free history category and demonstrate that conventional state semantics arises as a quotient of this richer history-based representation. In this formulation observable states appear as compressed summaries of the commitment histories that generated them, and the collapse functor is shown to be left adjoint to the history embedding.

Together these results provide a compact mathematical framework for systems whose natural semantics is append-only and history-preserving, including event-sourced architectures, version control systems, causal process models, and many forms of practical decision making.

1.2 Outline

This essay develops the formal structure underlying these contributions and accompanies each stage of the formalism with intuitive analogies. Section 2 situates the work relative to existing literature. Section 3 establishes notation. Section 4 presents the central idea in ordinary language. Section 5 reviews traced monoidal categories and isolates precisely what the trace operator discards. Section 6 introduces the Spherepop state representation. Section 7 defines the three primitive operators. Section 8 establishes their algebraic properties. Section 9 proves minimality of the operator set. Section 10 works through a minimal example. Section 11 establishes the representation theorem for deterministic computation. Section 12 characterizes state semantics as a quotient of history semantics. Section 13 describes the history algebra. Section 14 gives the categorical treatment. Section 15 establishes the collapse-history adjunction. Sections 16 through 22 draw connections to concurrency, entropy, programming practice, and reasoning. Section 23 discusses limitations. Section 24 draws the principal conclusions.

2 Related Work

The Spherepop framework intersects several existing research traditions concerned with the structure of computation, causality, and state evolution.

Categorical semantics of computation. Category theory has long provided a language for describing the structure of computational processes. The traced monoidal categories introduced by Joyal, Street, and Verity [2] give a categorical account of feedback and recursion in monoidal settings. In traced semantics internal loops may be collapsed into morphisms representing the overall input-output behavior of a system. Spherepop differs from traced semantics in its treatment of internal computation: rather than identifying morphisms that exhibit the same external behavior, the framework developed here preserves the ordered sequence of commitments that produced the result. Traced semantics may therefore be understood as a quotient of the richer history structure considered in this paper.

Concurrency and event structures. The study of concurrent computation has produced several models in which system behavior is represented by partially ordered events rather than sequential state transitions [4]. The generalization of Spherepop histories to partially ordered structures, discussed in Section 16, places the framework in close conceptual proximity to these models.

Distributed systems and append-only logs. Event-sourced architectures [6] reconstruct state by replaying a sequence of recorded events, while distributed databases often maintain conflict-free replicated data types whose histories converge through causal merging [8]. Version control systems such as Git record software evolution as a directed acyclic graph of commits [9]. These systems share a structural property central to Spherepop: the observable state is derived from a history of irreversible events.

Causal models in physics. Causal set theory models spacetime as a partially ordered set of events whose order encodes causal precedence [10]. Although the present work is concerned with computational semantics rather than physical ontology, the structural similarity highlights the generality of history-based descriptions.

Position of the present work. The contribution of this paper is to isolate a minimal algebraic framework that captures the append-only structure common to these domains. By introducing a simple state representation and three primitive operators, the framework provides a compact language for describing processes whose identity is determined by the ordered history of commitments through which they evolve.

3 Notation and Conventions

This section summarizes the principal symbols and conventions used throughout the paper.

The symbol Ω denotes a finite *option space* whose elements represent the possible commitments available to a Spherepop process. At time t the set of admissible possibilities is written $\Omega_t \subseteq \Omega$ and is called the *residual option space.*

Commitment histories are finite ordered sequences of elements of Ω. A history is written $H = [x_1, \ldots, x_n]$ where $x_i \in \Omega$ represents the ith commitment. The empty history is denoted $\varnothing$. When two histories are concatenated we write $H_1 \cdot H_2$.

The three primitive operators are written Pop, Bind, and Collapse. The operator $\textsc{Pop}_x$ represents an irreversible commitment to element $x \in \Omega_t$. The operator $\textsc{Bind}_f$ represents a contextual restriction by an admissibility function f. The operator Collapse reconstructs observable state from the commitment history.

Observable states belong to a category $\mathcal{S}$ of states and state transformations. The collapse operation induces a functor $\textsc{Collapse} : \mathcal{H}(\Omega) \to \mathcal{S}$ from the free history category to the category of observable states. All option spaces are assumed finite unless otherwise stated.

4 Why Histories Matter: The Central Idea

Before turning to the formal mathematics it is useful to state the central idea of Spherepop in ordinary language and to give the reader a collection of concrete images to hold in mind while the formalism is developed.

4.1 Programs as Functions vs. Programs as Histories

Most computational models treat a program as a function. You provide an input, the program runs, and eventually it produces an output. Once the output is known, the intermediate steps that produced it are usually discarded. They were merely a means to an end.

Spherepop treats those intermediate steps differently. Instead of discarding them, it records them permanently. The sequence of steps becomes the primary object of study.

The same contrast appears outside computing. If you edit a document repeatedly and keep only the final version, the intermediate drafts disappear. If instead you use a version control system, every change is recorded in a commit log. The identity of the document is no longer just the final text; it is the entire sequence of edits that produced it. Spherepop adopts the second perspective.

4.2 The Restaurant Analogy

Another illustration comes from everyday decision making. Imagine choosing a restaurant for dinner. The final choice may be "Italian." But the path to that choice matters. Perhaps Mexican was ruled out because the restaurant was closed, and sushi was ruled out because a friend dislikes it. Two evenings may end at the same restaurant yet involve entirely different chains of reasoning. In Spherepop those chains are preserved: what was eliminated, in what order, and by what constraint.

4.3 The Bubble Analogy

A visual way to picture the Spherepop state is to imagine a row of labeled bubbles floating in front of you. Each bubble represents a possible choice. As the computation proceeds, bubbles are popped one at a time. A popped bubble cannot be restored. The row of remaining bubbles shrinks, while the sequence of popped bubbles grows and is preserved as the record of what happened. The name “Spherepop” comes from this image.

4.4 Visual Summary of the Two Perspectives

The following two diagrams contrast the trace perspective with the Spherepop perspective.

$$\begin{tikzcd}
\boxed{\text{Input } A} \arrow[r] & f \arrow[r] & \boxed{\text{Output } B}
\end{tikzcd}$$

Internal computation
collapsed to a single morphism

Figure 1: Trace-style semantics. Internal computation is abstracted away and represented only by its overall input-output behavior.

$$\begin{tikzcd}
\boxed{\text{Initial State}} \arrow[r] & x_1 \arrow[r] & x_2 \arrow[r] & x_3 \arrow[r] & \boxed{\text{Observable State}}
\end{tikzcd}$$

Commitment history $H = [x_1, x_2, x_3]$

Figure 2: Spherepop semantics. The computation is represented by the ordered history of commitments that produced the observable state.

5 The Trace Operator and What It Hides

Before examining what Spherepop preserves, it is useful to be precise about what the trace operator discards.

5.1 What the Trace Does

The trace can be understood informally as a formal way of saying that the internal workings of a system are irrelevant once its external behavior is known. Trace semantics allows the program to contain internal loops, recursive calls, and intermediate computations, yet collapses all of those details into a single morphism $f : A \to B$.

Film analogy. Imagine watching a film and then writing a one-sentence plot summary. The summary may capture the overall outcome of the story but does not contain the sequence of scenes that produced it. Trace semantics treats the program like the summary. Spherepop treats it like the full film.

Definition 5.1 (Traced monoidal category)**.** *A* traced symmetric monoidal category *is a symmetric monoidal category* $(\mathcal{C}, \otimes, I)$ *equipped with a family of functions*

$$\mathrm{Tr}^U_{A,B} : \mathcal{C}(A \otimes U,\, B \otimes U) \;\longrightarrow\; \mathcal{C}(A, B),$$

one for each triple of objects A, B, U*, satisfying naturality, dinaturality, vanishing, and the yanking equation* $\mathrm{Tr}^U_{U,U}(\sigma_{U,U}) = \mathrm{id}_U$*.*

The dinaturality condition ensures that the trace is insensitive to the choice of representation for the internal computation: any two morphisms related by an internal change of type produce equal traces.

5.2 What the Trace Forgets

The trace performs a specific act of forgetting. It discards the ordered chain of intermediate steps that produced the output. Two programs that produce the same external function are identified as the same morphism regardless of their internal histories. In categorical language, trace semantics constructs a quotient that identifies morphisms whose internal behavior differs but whose external effect is the same. Spherepop declines that quotient and preserves the histories themselves.

Remark 5.1. *Consider two recursive programs that both compute the identity function* $A \to A$*. One returns immediately; the other performs a thousand recursive calls that cancel before returning. Under traced monoidal semantics both programs denote the same morphism* id_A*. Under Spherepop semantics the programs differ because the commitment histories differ.*

6 State Representation: Option Space and History

The primitive data structure of Spherepop is a pair representing the current state of an unfolding process.

Definition 6.1 (Spherepop state)**.** *Let* Ω *be a finite set called the* option space*. A* Spherepop state *is a pair*

$$(\Omega_t,\, H_t)$$

where $\Omega_t \subseteq \Omega$ *is the* residual option space *at time* t *and* $H_t = [x_1, x_2, \ldots, x_k]$ *is the* commitment history*, an ordered list of elements that have been selected and recorded.*

The initial state is $(\Omega, \varnothing)$. As the computation proceeds, Ω_t shrinks monotonically and H_t grows monotonically. These two monotonicity conditions are the central structural constraints of the framework.

The option space Ω_t records what *could have happened but did not*; the history H_t records what *did happen*. The asymmetry between them is intentional: the future is an unordered cloud of possibilities while the past is an ordered causal chain.

Example 6.1. *Let* $\Omega = \{a, b, c\}$. *After committing to* a *the state becomes* $(\{b, c\}, [a])$. *After committing to* b *it becomes* $(\{c\}, [a, b])$. *An alternative computation might produce* $(\{c\}, [b, a])$. *Both states share the same residual option space, but their histories are distinct ordered lists. In Spherepop these are distinct processes.*

7 The Three Primitive Operators

Three operators act on Spherepop states and constitute the complete operational vocabulary of the framework.

7.1 Pop: Irreversible Commitment

The bubble image. Pop is the act of popping one of the floating bubbles. The bubble disappears from the row and its label is appended to the record. The action cannot be undone.

Definition 7.1 (Pop)**.** *Let* (Ω_t, H_t) *be a Spherepop state and let* $x \in \Omega_t$. *The operator* $\textsc{Pop}_x$ *acts as*

$$\textsc{Pop}_x(\Omega_t,\ H_t) \ = \ \big(\Omega_t \setminus \{x\},\ \ H_t \cdot x\big).$$

The element x is removed from Ω_t and appended to H_t simultaneously. The operation is irreversible: there is no $\textsc{Pop}^{-1}$ that restores x to the option space and removes it from the history.

7.2 Bind: Contextual Restriction

The menu analogy. Before choosing a dish you filter the menu by dietary restriction. This filtering removes options without constituting a choice. Bind is that filter. Eliminated options leave no trace in the decision history because they were never genuine candidates.

Definition 7.2 (Bind). *Let $f : 2^\Omega \to 2^\Omega$ satisfy $f(\Omega_t) \subseteq \Omega_t$ for all Ω_t. The operator* $\textsc{Bind}_f$ *acts as*

$$\textsc{Bind}_f(\Omega_t,\, H_t) \;=\; \big(f(\Omega_t),\;\; H_t\big).$$

The history is unchanged because no commitment has occurred. Elements in $\Omega_t \setminus f(\Omega_t)$ are silently excluded. The distinction between exclusion by Bind and exclusion by Pop encodes the difference between structural impossibility and chosen commitment.

7.3 Collapse: Reconstruction from History

The ledger analogy. A bank account stores a ledger of deposits and withdrawals rather than a fixed balance. The current balance is computed by replaying the ledger. Collapse is that replay operation.

Definition 7.3 (Collapse). *Let s_0 be a fixed root state and let $\{T_x\}_{x\in\Omega}$ be a family of state transformations. For a history $H_t = [x_1, x_2, \ldots, x_n]$, the operator* Collapse *acts as*

$$\textsc{Collapse}(H_t) \;=\; T_{x_n} \circ T_{x_{n-1}} \circ \cdots \circ T_{x_1}(s_0).$$

The observable state is derived by composing transformations in temporal order. In general $T_a \circ T_b \neq T_b \circ T_a$, so histories $[a, b]$ and $[b, a]$ may yield different observable states. The history must therefore be an ordered list rather than a set.

8 Algebraic Properties of the Spherepop Operators

The operators exhibit several structural properties that follow directly from the monotonic nature of commitment histories.

Proposition 8.1 (Monotonicity of option spaces). *Let (Ω_t, H_t) be a Spherepop state. After any sequence of operations the option space satisfies $\Omega_{t+1} \subseteq \Omega_t$.*

Proof. Both Pop and Bind replace Ω_t with a subset of itself: $\textsc{Pop}_x$ removes one element, while $\textsc{Bind}_f$ replaces Ω_t with $f(\Omega_t) \subseteq \Omega_t$ by definition. □

Proposition 8.2 (History extension). *For every Spherepop computation the commitment history satisfies $H_t \preceq H_{t+1}$, where $\preceq$ denotes the prefix relation on sequences.*

Proof. Bind leaves the history unchanged, while Pop appends one element. In either case H_{t+1} contains H_t as a prefix. □

Proposition 8.3 (Non-commutativity of commitments). *For $a, b \in \Omega$ with $a \neq b$, in general* $\textsc{Pop}_a \circ \textsc{Pop}_b \neq \textsc{Pop}_b \circ \textsc{Pop}_a$.

Proof. $\textsc{Pop}_a$ followed by $\textsc{Pop}_b$ produces history $[a, b]$, while the reverse order produces $[b, a]$. Since histories are ordered lists these sequences are distinct whenever $a \neq b$. □

Non-commutativity is essential: if commitments commuted, the causal ordering of events would disappear from the model.

Proposition 8.4 (Collapse as a functor). *The operator* Collapse *defines a functor from the free history category $\mathcal{H}(\Omega)$ to the category of observable states.*

Proof. For histories H_1 and H_2, $\textsc{Collapse}(H_1 \cdot H_2) = \textsc{Collapse}(H_2) \circ \textsc{Collapse}(H_1)$. Concatenation of histories therefore corresponds to composition of state transformations. Identity histories map to identity transformations, establishing functoriality. □

9 Minimality of the Spherepop Operator Set

It is natural to ask whether the operator set $\{\textsc{Pop}, \textsc{Bind}, \textsc{Collapse}\}$ is minimal. This section proves that it is: the three operators are jointly sufficient to generate the history algebra, and no proper subset recovers the full semantics.

Proposition 9.1 (Sufficiency). *Every Spherepop computation over a finite option space Ω can be expressed as a composition of* Bind *and* Pop *operations followed by a final application of* Collapse*.*

Proof. Any change to the residual option space that records no commitment is by definition an admissibility restriction and hence a Bind operation. Any change that both removes an element from the option space and records it in the history is by definition a Pop operation. Since the history grows only by appending and the option space evolves only by restriction, any finite computation decomposes into a finite composition of such operations. The observable state is then obtained by Collapse. □

Proposition 9.2 (Necessity of Pop). *No system generated solely by* Bind *and* Collapse *can recover the commitment-sensitive semantics of Spherepop.*

Proof. Bind restricts the option space but leaves the history unchanged. Collapse reconstructs state from an existing history but creates no new commitments. A system lacking Pop cannot extend the history and cannot distinguish a structurally excluded possibility from a genuinely chosen one. The defining irreversibility of Spherepop is therefore absent. □

Proposition 9.3 (Necessity of Bind)**.** *No system generated solely by* Pop *and* Collapse *can represent the distinction between structural exclusion and chosen commitment.*

Proof. Pop always appends an element to the history, recording it as an actual commitment. Spherepop requires the ability to restrict the option space without recording anything in the history. Without Bind, every elimination of a possibility would be forced to appear as a commitment, and the difference between environmental constraint and internal choice would disappear. □

Proposition 9.4 (Necessity of Collapse)**.** *No system generated solely by* Pop *and* Bind *can recover the observable state semantics of Spherepop.*

Proof. Pop and Bind determine how the history and option space evolve, but do not specify how the resulting history is interpreted as an observable state. Without Collapse, the system has commitment dynamics but no reconstruction semantics. □

Theorem 9.1 (Minimality of the Spherepop operator set)**.** *The operator set* {Pop, Bind, Collapse} *is minimal: it is sufficient to generate the history algebra, and no proper subset recovers both the commitment dynamics and the observable reconstruction semantics.*

Proof. Sufficiency is established by the first proposition. The three necessity propositions show that removing any one operator destroys an essential aspect of the framework. □

10 A Minimal Spherepop Program

We now trace through the smallest nontrivial Spherepop computation. Let $\Omega = \{a, b, c\}$, initial state $(\{a, b, c\},\, \varnothing)$, and root state s_0.

Step 0. $(\{a, b, c\},\, \varnothing)$. All possibilities available, no commitments made.

Step 1: Apply Bind$_f$ with $f(\Omega_t) = \Omega_t \setminus \{c\}$.

$$\big(\{a, b\},\, \varnothing\big).$$

The element c is excluded by context. The history remains empty: no commitment has occurred.

Step 2: Apply Pop$_a$.

$$\big(\{b\},\, [a]\big).$$

The history records that a was the first commitment.

Step 3: Apply Collapse($[a]$).

$$s \;=\; T_a(s_0).$$

A traditional functional model returns $T_a(s_0)$ and discards the intermediate structure. Spherepop preserves the pair $(\{b\},\, [a])$. The residual b records that a genuine alternative was available at the moment of commitment. The identity of the computation is this pair, not merely its output.

Two computations both producing $s = T_a(s_0)$ may differ in provenance: one may have applied $\textsc{Bind}_f$ before $\textsc{Pop}_a$, leaving state $(\{b\},\, [a])$; another may have applied only $\textsc{Pop}_a$ with no prior restriction, leaving $(\{b, c\},\, [a])$. In Spherepop these are genuinely different objects.

11 Deterministic Computation as History

The relationship between Spherepop semantics and conventional state-based computation can be made precise.

11.1 Encoding Transitions as Commitments

Suppose a deterministic system evolves according to $s_{t+1} = F(s_t)$. Each transition may be associated with a symbolic commitment $x_t \in \Omega$ indexing the possible transition types. The execution therefore generates a history $H = [x_1, \dots, x_n]$. If each x_i corresponds to a transformation T_{x_i} acting on states, the observable state is

$$s_n = T_{x_n} \circ \cdots \circ T_{x_1}(s_0) = \textsc{Collapse}(H).$$

This is precisely the $\textsc{Collapse}$ operator.

11.2 Representation Theorem

Theorem 11.1 (Deterministic computation as history)**.** *Every deterministic computation expressible as a sequence of state transitions admits a representation as a Spherepop history whose collapse reproduces the observable state evolution.*

Proof. Let the computation be $s_0 \to s_1 \to \cdots \to s_n$. Associate to each transition $s_{i-1} \to s_i$ a symbol $x_i \in \Omega$ and define T_{x_i} satisfying $T_{x_i}(s_{i-1}) = s_i$. The transition sequence corresponds to $H = [x_1, \dots, x_n]$. Applying $\textsc{Collapse}$ yields $T_{x_n} \circ \cdots \circ T_{x_1}(s_0) = s_n$. □

State-based semantics may therefore be understood as a compressed description of the underlying commitment history. Spherepop restores the structure that this compression discards.

12 State Semantics as a Quotient of History Semantics

The relationship between Spherepop semantics and conventional state-based semantics can be stated precisely in algebraic terms.

Definition 12.1 (Observable equivalence)**.** *Two histories $H, H' \in \Omega^*$ are* observationally equivalent *if*

$$\textsc{Collapse}(H) = \textsc{Collapse}(H').$$

This relation is written $H \sim H'$.

The quotient set

$$\Omega^* / \sim$$

collects equivalence classes of histories that produce the same observable state. This quotient construction recovers the familiar state-based semantics: instead of distinguishing histories, the system identifies them whenever they produce the same state.

Loss of causal structure. Histories $[a, b]$ and $[b, a]$ may collapse to the same observable state when $T_a \circ T_b = T_b \circ T_a$. State semantics identifies these histories; Spherepop does not.

The quotient map

$$\pi : \Omega^* \longrightarrow \Omega^* / \sim$$

forgets the internal sequence of commitments and retains only the observable result. Spherepop declines this compression and retains the full causal structure. Conventional state semantics therefore arises as the observable projection of the richer history semantics: it is Spherepop quotiented by observable equivalence.

13 The History Algebra

The operators of the previous sections generate the *history algebra.* The key observation is that commitment histories behave exactly like words in a free monoid.

The collection of all finite ordered sequences over Ω forms the free monoid Ω^* under concatenation. The empty sequence $\varnothing$ is the identity element. Concatenation is associative but not commutative: $[a, b] \neq [b, a]$ in general. There are no further relations. Distinct sequences remain distinct; the free monoid imposes no equations identifying different words.

Definition 13.1 (History algebra)**.** *A* history algebra *over a finite option space Ω consists of the set of all Spherepop states $\{(\Omega_t, H_t) \mid \Omega_t \subseteq \Omega,\ H_t \in \Omega^*\}$; for each $x \in \Omega_t$ the operator* $\textsc{Pop}_x$*; for each admissibility function f the operator* $\textsc{Bind}_f$*; and a root state s_0 together with a family $\{T_x\}_{x\in\Omega}$ defining* $\textsc{Collapse}$*.*

The absence of imposed relations between distinct commitment sequences is a design principle rather than an omission. By refraining from quotienting histories that produce the same observable state, the history algebra preserves the full causal provenance of each computation.

Git analogy. Applying commit A then commit B differs from applying B then A whenever the commits modify overlapping files. The non-commutativity of the history algebra is the same non-commutativity that makes merge order matter in version control.

14 The Free History Category

The history algebra admits a natural categorical interpretation.

Definition 14.1 (Free history category)**.** *The* free history category $\mathcal{H}(\Omega)$ *over a finite option space Ω is the category whose objects are Spherepop states (Ω_t, H); whose morphisms from (Ω_t, H) to (Ω_t', H') are pairs (σ, τ) where $\sigma : \Omega_t \to \Omega_t'$ is a set map and $\tau \in \Omega^*$ satisfies $H' = H \cdot \tau$; whose composition is $(\sigma', \tau') \circ (\sigma, \tau) = (\sigma' \circ \sigma,\ \tau \cdot \tau')$; and whose identity morphisms are $(\mathrm{id}_{\Omega_t}, \varnothing)$.*

Proposition 14.1. *The category $\mathcal{H}(\Omega)$ is well-defined. The* $\textsc{Pop}$ *and* $\textsc{Bind}$ *operators correspond to generating morphisms in $\mathcal{H}(\Omega)$, and* $\textsc{Collapse}$ *is a functor $\mathcal{H}(\Omega) \to \mathcal{S}$.*

Proof. Associativity of composition follows from associativity of concatenation and function composition. Identity morphisms satisfy the required equations because appending the empty sequence leaves any history unchanged. The functor $\textsc{Collapse}$ sends $[x_1, \ldots, x_n]$ to $T_{x_n} \circ \cdots \circ T_{x_1}$; functoriality follows because concatenation corresponds to composition of transformations. $\square$

The category $\mathcal{H}(\Omega)$ is called free because it introduces no relations between histories. Each distinct sequence of commitments corresponds to a distinct morphism. This is the categorical analogue of the free monoid: the object generated by a set of generators with no imposed equations. Traced monoidal categories construct quotients; Spherepop constructs a free object.

Remark 14.1 (Path objects)**.** *The morphisms of* $\mathcal{H}(\Omega)$ *are path objects: they are the paths themselves, not equivalence classes. Their identity is literal equality of ordered sequences. This is why Spherepop computations feel structurally similar to event-sourcing logs and git commit graphs—the mathematics is the same idea expressed at different scales.*

15 The Collapse Functor and History Embedding

The relationship between history-based semantics and conventional state semantics can be expressed naturally in categorical terms through an adjunction.

Proposition 15.1. *The operator* $\textsc{Collapse}$ *defines a functor* $\textsc{Collapse} : \mathcal{H}(\Omega) \to \mathcal{S}$*.*

Proof. For histories H_1 and H_2, $\textsc{Collapse}(H_1 \cdot H_2) = \textsc{Collapse}(H_2) \circ \textsc{Collapse}(H_1)$. Concatenation of histories corresponds to composition of state transformations; the empty history maps to the identity transformation. □

Conversely, any observable state may be embedded into the history category by associating it with the trivial history that produces it.

Definition 15.1 (History embedding)**.** *Define a functor* $\mathrm{Embed} : \mathcal{S} \to \mathcal{H}(\Omega)$ *that maps each state to the corresponding history representation, treating a state as the degenerate history whose collapse yields that state.*

Theorem 15.1 (Collapse-history adjunction)**.** *The collapse functor* $\textsc{Collapse}$ *is left adjoint to the history embedding functor* Embed*.*

Proof. For any history object H in $\mathcal{H}(\Omega)$ and any state s in $\mathcal{S}$ there is a natural correspondence

$$\mathrm{Hom}_{\mathcal{S}}\big(\textsc{Collapse}(H),\, s\big) \;\cong\; \mathrm{Hom}_{\mathcal{H}(\Omega)}\big(H,\, \mathrm{Embed}(s)\big).$$

A morphism from the collapsed state to s corresponds to a history extension transforming H into a history whose collapse yields s. This bijection respects composition and identity, establishing the adjunction. □

This adjunction expresses the conceptual relationship precisely. The collapse functor maps histories to their observable states, discarding internal commitment structure. The embedding functor performs the reverse by treating a state as the trivial history that produces it. States are therefore compressed summaries of the histories that generated them—a fact made precise by the universal property of the adjunction.

16 Partial Orders of Commitment

The framework developed so far models commitment histories as ordered lists, appropriate for sequential systems. Many real systems exhibit concurrency: multiple events may occur independently without a fixed global ordering.

A more general representation replaces linear sequences with partially ordered sets of events. Let E be a set of commitment events and let $\prec$ be a partial order on E representing causal precedence. The pair $(E, \prec)$ forms a causal history. The relation $e_1 \prec e_2$ indicates that e_1 must precede e_2; incomparable events may occur independently. Linear histories appear as the special case where $\prec$ is a total order.

Partially ordered histories may be visualized as directed acyclic graphs, closely resembling the commit graphs of distributed version control systems. Lamport's logical clocks and vector clocks were introduced precisely to track such causal relationships in distributed computation; Spherepop can be viewed as the sequential special case of a more general causal history framework.

The Collapse operator extends naturally to partially ordered histories: instead of composing transformations along a linear sequence, collapse composes them along any topological ordering consistent with $\prec$. Because incomparable events have no causal dependency, their associated transformations commute with respect to observable reconstruction, so any topological ordering produces the same final observable state.

The append-only property of histories remains unchanged: new events extend the causal graph without modifying existing structure. The identity of the process therefore remains the history itself, whether linear or partially ordered.

17 Entropy and Irreversibility in Spherepop Systems

The defining feature of Spherepop computations is the irreversible elimination of possibilities. This monotonic structure suggests a natural interpretation in terms of information theory.

17.1 Option Space as Information

If the system initially admits $|\Omega|$ possible alternatives, the uncertainty associated with the option space can be measured by

$$S_t = \log |\Omega_t|.$$

This quantity represents the amount of information required to specify which element will ultimately be selected.

17.2 Commitment as Entropy Reduction

Each application of Pop removes one element from the option space:

$$|\Omega_{t+1}| < |\Omega_t|, \qquad S_{t+1} < S_t.$$

The Spherepop process therefore describes a trajectory of progressive information gain. Each commitment reduces uncertainty about the final configuration.

17.3 History as Information Reservoir

Because excluded possibilities cannot be recovered, the commitment history plays the role of an informational reservoir. It preserves the causal sequence of decisions that produced the present configuration. Without the history, the system retains only the final state, and the intermediate structure is lost. The history compensates for the information destroyed by irreversible commitments.

A Spherepop computation can therefore be understood as a trajectory through a space of possibilities in which entropy steadily decreases as commitments accumulate. This interpretation aligns naturally with systems governed by genuine irreversibility and connects the Spherepop framework to the broader thermodynamic picture of computation, in which irreversible operations incur a minimum physical cost measured by Landauer's principle.

18 Connections to Append-Only Systems

The mathematical structure identified in the preceding sections appears under different names in several domains. These are not analogies imposed from outside; they are instances of the same algebraic structure.

Event sourcing. In event-sourced architectures the state is derived from an append-only log of events by replaying them from the initial configuration [6]. This is precisely the Collapse operator: $\textsc{Collapse}([x_1, \ldots, x_n]) = T_{x_n} \circ \cdots \circ T_{x_1}(s_0)$.

Version control. A git repository is an append-only directed acyclic graph of commits [9]. Pop corresponds to committing; Bind corresponds to pre-commit hooks that restrict admissible changes; Collapse corresponds to checking out a branch. The non-commutativity of Spherepop histories corresponds to the non-commutativity of git merges.

Ledgers and blockchains. Financial ledgers and blockchains enforce append-only transaction sequences, with the current state reconstructed by replaying the ledger [7]. Both are instances of the free history category $\mathcal{H}(\Omega)$.

Causal sets in physics. In causal set theory, spacetime is a locally finite partial order of events whose order encodes causal precedence [10]. The history of a physical system is derived by composing the effects of those events in causal order—formally identical to the COLLAPSE operator.

19 Spherepop Patterns in Programming Practice

The Spherepop pattern is already familiar to programmers, appearing in scope resolution, lazy evaluation, and the treatment of side effects.

Scope resolution. When an interpreter encounters a symbol, it searches through nested environments from innermost to outermost. Each scope check acts as a BIND-like restriction, reducing the set of possible interpretations. When the correct binding is identified, the selection corresponds to POP. Evaluating the associated expression corresponds to COLLAPSE. Scope resolution is therefore an instance of the Spherepop pattern: a search through nested domains that progressively restricts possibilities until a commitment is made.

Lazy evaluation as deferred collapse. In lazy languages, expressions are not evaluated until their value is required. From the Spherepop perspective this corresponds to delaying the application of COLLAPSE. The program accumulates a structure describing possible transformations without committing to their concrete execution. Only when the observable result is demanded does the system collapse the accumulated structure.

Side effects as commitments. Operations that produce side effects—writing to disk, mutating shared state, sending a network message—behave like POP: once executed, they alter the external state irreversibly. Pure functional programming isolates such operations at the boundary of the system, preserving the freedom of internal computation until commitment is unavoidable. This programming discipline can be understood as maintaining optionality—keeping the option space large for as long as possible—which is precisely the orientation motivating the Spherepop framework.

20 Nested Domains and Problem Solving

The Spherepop pattern appears in human problem solving more broadly. When confronting a complex task, individuals rarely explore the entire space of possibilities simultaneously. Instead they work through a sequence of nested constraints, ruling out options by context and resources before making a commitment.

The evolving situation is naturally represented as the pair (Ω_t, H_t), where Ω_t represents viable remaining options and H_t records commitments already made. Each new decision reduces Ω_t while extending H_t. The final outcome is the cumulative effect of the entire commitment history, not a single decisive step.

21 Reactive Pipelines in Practical Activity

Many practical activities follow a reactive pipeline structure naturally described in Spherepop terms.

Painting. Each brushstroke restricts what can be painted next. Once pigment has been applied the previous state cannot be restored without leaving evidence. The completed painting is the result of a long chain of irreversible commitments.

Cooking. Preparation steps progressively restrict possibilities. Once ingredients are combined or cooked the previous configuration cannot be recovered. The finished dish is the collapse of the transformation history.

Construction. Early design choices restrict later possibilities. Structural decisions about foundations and layout eliminate entire branches of potential designs. The final building is not simply a static object but the cumulative result of a commitment sequence made during construction.

Across all these domains the underlying structure is the same: a system begins with a broad space of possibilities and evolves through a sequence of irreversible commitments whose cumulative effect produces the observable outcome.

22 Spherepop and Human Cognition

The Spherepop framework provides a natural language for describing aspects of human cognition. Reasoning processes frequently unfold as a progressive narrowing of possibilities rather than as a direct evaluation of a predetermined function.

When humans attempt to understand a situation they explore a space of candidate interpretations, gradually eliminating those inconsistent with available evidence. Each new piece of information acts as a Bind-like restriction. A decision or inference performs a Pop, selecting one interpretation and recording it in the evolving reasoning history. The resulting mental model corresponds to Collapse applied to the accumulated commitments.

The reasoning process cannot be reduced to its final conclusion. Two individuals may reach the same answer while following entirely different chains of inference. In scientific explanations, legal arguments, and engineering designs the reasoning history matters: the conclusion acquires its justification from the ordered sequence of steps that produced it.

Cognitive science increasingly recognizes that human reasoning is path dependent. Early assumptions constrain later interpretations, and the order in which information is encountered can influence the final outcome. Spherepop provides a concise mathematical description of this phenomenon: cognition unfolds as an append-only history of commitments within a dynamically shrinking space of possibilities.

23 Limitations and Scope of the Framework

The Spherepop framework is intentionally minimal, and several limitations of the present formulation should be noted.

Discrete commitments. The model assumes that system evolution decomposes into discrete commitment events drawn from a finite option space. Many real-world systems exhibit continuous dynamics. While such systems can often be approximated by sufficiently fine discretization, the present framework does not model continuous processes directly.

Finite option spaces. The option space has been treated as finite throughout. Extensions to countably infinite or dynamically generated option spaces are possible in principle but require additional technical machinery.

Sequential commitment. The basic model represents histories as totally ordered sequences. Section 16 showed how this generalizes to partially ordered causal graphs for concurrent systems, but a fully general concurrent Spherepop theory remains an open direction.

Reversible computation. Spherepop is designed for processes governed by irreversible commitments. Reversible computational models, in which the system retains sufficient information to reconstruct prior states exactly, do not naturally exhibit append-only histories. The Spherepop framework complements rather than replaces reversible computation models.

Abstraction level. The framework operates at a deliberately abstract level. The option space Ω, the transformation family $\{T_x\}$, and the admissibility functions used by BIND are treated as primitives. In concrete applications these objects must be specified by the surrounding domain theory. The goal of the present work is to clarify the algebraic role played by commitment histories once they exist, not to prescribe specific domain structures.

24 Conclusion

Many important systems in computation, engineering, and cognition do not behave like pure functions whose identity is determined solely by their input-output relation. Instead they behave like histories: ordered sequences of irreversible commitments whose cumulative effect produces the observable state of the system.

Traditional semantic frameworks, including traced monoidal categories, achieve their power by abstracting over internal computation. The trace operator collapses feedback loops and identifies morphisms that exhibit the same external behavior regardless of internal path. For many purposes this abstraction is both elegant and appropriate. But the systems examined in this paper belong to a different structural class. In these systems the internal path cannot be discarded without losing essential information.

Spherepop addresses this class by preserving the commitment history rather than collapsing it. The state is represented as the pair (Ω_t, H_t), recording both what remains possible and what has already been committed. Three primitive operators govern evolution: BIND restricts the option space without commitment; POP performs an irreversible commitment and records it; COLLAPSE derives the observable state by composing transformations in temporal order. The minimality theorem establishes that this operator set is tight: no proper subset recovers the full semantics.

The resulting structure is the free history category $\mathcal{H}(\Omega)$, whose morphisms correspond to elements of the free monoid Ω^*. No two distinct histories are identified. Where traced monoidal semantics constructs a quotient, Spherepop constructs a free object. The collapse-history adjunction makes this duality precise categorically: conventional state semantics appears as the image of the collapse functor, and the adjunction identifies the exact sense in which states are compressed summaries of the histories that generated them.

This structure appears naturally across many domains: event-sourced architectures, version control systems, financial ledgers, blockchains, scope resolution and lazy evaluation in programming languages, iterative problem solving, and the path-dependent character of human reasoning. In all of these the same structural pattern recurs: a system begins with a space of possibilities, evolves through contextual restrictions and irreversible commitments, and produces observable states by composing the effects of those commitments in temporal order.

History before state. The perspective developed in this work may be summarized by a simple inversion of the usual semantic hierarchy. Traditional models begin with states and treat histories as auxiliary records that may be discarded once the observable outcome is known. In Spherepop the relationship is reversed: histories are primary and states arise as their observable shadows. The collapse functor maps append-only commitment structures to conventional state semantics, while traced monoidal categories represent the opposite tendency to identify internal paths once their external effect is determined. In practice many real systems lie between these extremes. Event-sourced architectures preserve explicit commit logs; version control systems maintain directed histories of edits; and pure functional programming delays evaluation and side effects so that commitments occur only when necessary. Across these domains the same structural insight reappears: when processes evolve through irreversible decisions, the identity of the system resides not merely in its present state but in the causal history that produced it.

The appendices provide a constructive presentation of Spherepop. Appendix A introduces a concrete syntax for Spherepop expressions. Appendix B describes a simple typed algebra capturing the commitment operators and defines a category of constructions. Appendix C demonstrates that elementary logic gates can be constructed from the primitive operators. Appendix D treats memory devices and counters. Appendix E establishes that histories form a free monoid object. Appendix F shows that the free history category is the path category generated by that monoid.

A Abstract Syntax of Spherepop

We describe a minimal concrete syntax for Spherepop programs. Let Ω be a finite option space whose elements are written $a, b, c, \ldots$. Spherepop terms are generated by the grammar

$$\begin{array}{rcl} t & ::= & (\Omega_t, H_t) \\ & | & \textsc{Bind}_f(t) \\ & | & \textsc{Pop}_a(t) \\ & | & \textsc{Collapse}(t) \\ & | & t_1; t_2 \end{array}$$

where (Ω_t, H_t) denotes an explicit Spherepop state, $\textsc{Bind}_f$ denotes contextual restriction by admissibility function f, $\textsc{Pop}_a$ denotes irreversible commitment to $a \in \Omega_t$, $\textsc{Collapse}$ reconstructs observable state, and $t_1; t_2$ represents sequential composition. The operational reduction rules are

$$\textsc{Bind}_f(\Omega_t, H_t) \longrightarrow (f(\Omega_t), H_t),$$

$$\textsc{Pop}_a(\Omega_t, H_t) \longrightarrow (\Omega_t \setminus \{a\},\ H_t \cdot a) \qquad (a \in \Omega_t),$$

$$\textsc{Collapse}(\Omega_t, H_t) \longrightarrow T_{x_n} \circ \cdots \circ T_{x_1}(s_0) \qquad (H_t = [x_1, \ldots, x_n]).$$

The first two rules evolve internal Spherepop structure; the third interprets that structure as an observable value.

B Typed Spherepop Algebra

We refine the framework with a simple type discipline distinguishing option spaces, histories, states, and observable outputs.

B.1 Types and Typing Judgments

The basic types are Opt, Hist, State, and Obs, with

$$\mathsf{State} \;\cong\; \mathcal{P}_{\text{fin}}(\Omega) \times \Omega^*.$$

Typing judgments have the form $\Gamma \vdash t : A$. The principal rules are:

$$\frac{}{\Gamma \vdash (\Omega_t, H_t) : \mathsf{State}} \qquad \frac{\Gamma \vdash t : \mathsf{State}}{\Gamma \vdash \textsc{Bind}_f(t) : \mathsf{State}} \qquad \frac{\Gamma \vdash t : \mathsf{State} \quad a \in \Omega_t}{\Gamma \vdash \textsc{Pop}_a(t) : \mathsf{State}} \qquad \frac{\Gamma \vdash t : \mathsf{State}}{\Gamma \vdash \textsc{Collapse}(t) : \mathsf{Obs}}$$

B.2 Typed Algebra

A typed Spherepop algebra over a finite option space Ω consists of a carrier state object $S = \mathcal{P}_{\text{fin}}(\Omega) \times \Omega^*$, together with a family of admissibility endomorphisms $\textsc{Bind}_f : S \to S$, a family of partial commitment endomorphisms $\textsc{Pop}_a : S \rightharpoonup S$ for $a \in \Omega$, and a reconstruction map $\textsc{Collapse} : S \to O$ into an observable object O. In this sense Spherepop forms a typed algebra over a signature whose primitive operations are the families $\{\textsc{Bind}_f\}_f$, $\{\textsc{Pop}_a\}_{a\in\Omega}$, and the map $\textsc{Collapse}$.

B.3 A Category of Constructions

Define a category $\mathcal{C}_{\text{Sp}}$ of Spherepop constructions whose objects are typed interfaces (Ω, A) where Ω is an option space and A is a carrier of observable states, and whose morphisms $(\Omega, A) \to (\Omega', B)$ are well-typed Spherepop constructions taking states over Ω to states over Ω' with a compatible reconstruction into B. Composition is given by composition of constructions; identity morphisms leave both option space and history unchanged. This category makes precise the idea that Spherepop programs are structured constructions assembled from typed commitment operations.

C Logic Gates as Spherepop Constructions

Elementary logical devices may be represented in Spherepop form by interpreting commitment symbols as output assignments. Let the observable state space be $\mathsf{Bool} = \{0, 1\}$ and introduce commitment symbols `set0` and `set1` with associated transformations $T_{\texttt{set0}}(s) = 0$ and $T_{\texttt{set1}}(s) = 1$. Logical evaluation proceeds by restricting the option space to the admissible output commitment and then performing a single $\textsc{Pop}$.

For the NOT gate with input $b \in \{0, 1\}$ define $f_b(\Omega) = \{\texttt{set}(1 - b)\}$. The NOT gate is $\textsc{Bind}_{f_b}; \textsc{Pop}_{\texttt{set}(1-b)}$, and collapse yields $1 - b$.

For the AND gate with inputs p, q define $f_{p,q}(\Omega) = \{\texttt{set1}\}$ if $p = q = 1$ and $\{\texttt{set0}\}$ otherwise. The AND gate is $\textsc{Bind}_{f_{p,q}}; \textsc{Pop}_{\texttt{set}(p\wedge q)}$.

For the OR gate, $f_{p,q}(\Omega) = \{\texttt{set1}\}$ if $p = 1$ or $q = 1$, and $\{\texttt{set0}\}$ otherwise. For XOR, $f_{p,q}(\Omega) = \{\texttt{set1}\}$ if $p \neq q$ and $\{\texttt{set0}\}$ if $p = q$.

All Boolean gates follow the same pattern: admissibility restriction narrows the option space to the correct output symbol, and a single Pop commits to it. Even in these simplest cases the framework records the causal history of the decision rather than merely its extensional value.

D Memory Devices and Counters

Because Spherepop histories are append-only structures, counting operations arise naturally as functions of the commitment sequence.

Unary counter. Introduce a single commitment symbol `tick` and define $\textsc{Collapse}(\left[\underbrace{\texttt{tick}, \ldots, \texttt{tick}}_{n}\right]) =$ n. Incrementing the counter is simply $\textsc{Pop}_{\texttt{tick}}$.

Binary counter. Let $\mathsf{Bin}_k = \{0, 1\}^k$ be the observable state space and let $T_{\texttt{tick}}$ be the binary successor function modulo 2^k. For a history $H_n = [\texttt{tick}, \ldots, \texttt{tick}]$ of length n, collapse yields $\textsc{Collapse}(H_n) = T^n_{\texttt{tick}}(0, \ldots, 0)$.

One-bit latch. With commitment symbols `write0`, `write1`, `hold` and transformations $T_{\texttt{write0}}(m) = 0$, $T_{\texttt{write1}}(m) = 1$, $T_{\texttt{hold}}(m) = m$, a latch step uses Bind to restrict the admissible action based on control inputs and then performs the corresponding Pop. Collapse reconstructs the current bit by replaying the write history.

These constructions illustrate the general principle: Spherepop handles memory not by privileging mutable storage but by treating stored values as derived summaries of commitment history. Latches, counters, and other stateful devices are therefore historical constructions rather than self-subsistent state containers.

E Histories as a Free Monoid Object

The algebraic structure underlying Spherepop histories is characterized categorically as a free monoid object generated by the option space.

The set Ω^* of all finite ordered sequences over Ω carries a monoid structure under concatenation $\cdot$, with identity element $\varepsilon = [\,]$. Associativity follows from associativity of concatenation.

Proposition E.1 (Universal property of the history monoid)**.** *Let $(M, \cdot, e)$ be any monoid and let $\phi : \Omega \to M$ be a function. Then there exists a unique monoid homomorphism $\widehat{\phi} : \Omega^* \to M$ satisfying $\widehat{\phi}([a]) = \phi(a)$ for all $a \in \Omega$.*

Proof. Define $\widehat{\phi}([x_1, \ldots, x_n]) = \phi(x_1)\cdot\phi(x_2)\cdots\phi(x_n)$. This respects concatenation and sends ε to e. Uniqueness follows from the requirement that $\widehat{\phi}$ be a homomorphism extending ϕ. □

The collapse operator Collapse : $\Omega^* \to \mathcal{S}$ is therefore a monoid homomorphism determined by the assignment $a \mapsto T_a$. Observable states arise as the image of the free history monoid under this homomorphism. The Spherepop framework preserves the free monoid of commitment histories; traditional semantics collapses it to the quotient $\Omega^*/\sim$.

F The Free History Category as a Path Category

The free history category $\mathcal{H}(\Omega)$ is precisely the path category generated by the free monoid of histories.

Each $a \in \Omega$ may be viewed as a directed edge $* \xrightarrow{a} *$ in a directed graph with a single object $*$. Finite paths in this graph correspond to finite sequences $[x_1, \ldots, x_n] \in \Omega^*$, and composition of paths corresponds to concatenation. The resulting path category $\mathcal{P}(G)$ therefore has $\mathrm{Hom}(*, *) = \Omega^*$ with composition given by concatenation and identity morphism the empty path.

Spherepop commitment histories are exactly these paths. A history $H = [x_1, \ldots, x_n]$ corresponds to the composite path

$$* \xrightarrow{x_1} * \xrightarrow{x_2} \cdots \xrightarrow{x_n} *.$$

Appending a new commitment corresponds to extending the path by one additional edge. The append-only property corresponds to the categorical fact that paths extend by composition but cannot be shortened without changing the morphism.

The collapse operator defines a functor Collapse : $\mathcal{P}(G) \to \mathcal{S}$ determined by assigning each generating edge $a \in \Omega$ to the transformation $T_a : \mathcal{S} \to \mathcal{S}$. Functoriality ensures that a composite path is mapped to the corresponding composition of transformations.

The entire Spherepop semantics may therefore be summarized as a functor from the free path category generated by the option space into the category of observable states. This construction makes precise the sense in which Spherepop preserves the full path structure of commitment sequences rather than collapsing them into equivalence classes: the framework retains the free path category while conventional semantics takes its quotient by observable equivalence.

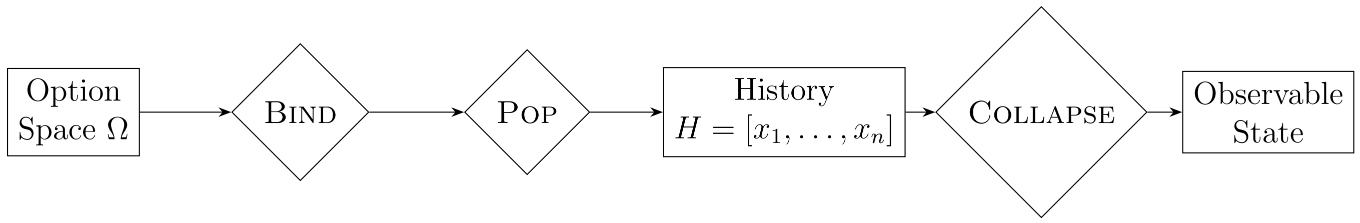

Figure 3: The Spherepop semantic pipeline. Contextual restriction (Bind) narrows the option space; irreversible commitment (Pop) records selections in the append-only history; and observable state is reconstructed by applying Collapse to the resulting commitment sequence. In this formulation observable states are derived objects; the primary semantic structure is the append-only history of commitments.

References

[1] Saunders Mac Lane. *Categories for the Working Mathematician.* 2nd edition. Springer, 1998.

[2] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[3] Samson Abramsky and Achim Jung. Domain theory. In *Handbook of Logic in Computer Science*, Volume 3. Oxford University Press, 1994.

[4] Glynn Winskel and Mogens Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, Volume 4. Oxford University Press, 1995.

[5] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[6] Martin Fowler. Event sourcing. `martinfowler.com`, 2005.

[7] Martin Kleppmann. *Designing Data-Intensive Applications.* O'Reilly Media, 2017.

[8] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2011.

[9] Scott Chacon and Ben Straub. *Pro Git*. Apress, 2014.

[10] Graham Brightwell and Rafael Sorkin. Structure of causal sets. *Physical Review D*, 44(4):1027–1037, 1991.

[11] John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. 3rd edition. Pearson, 2006.