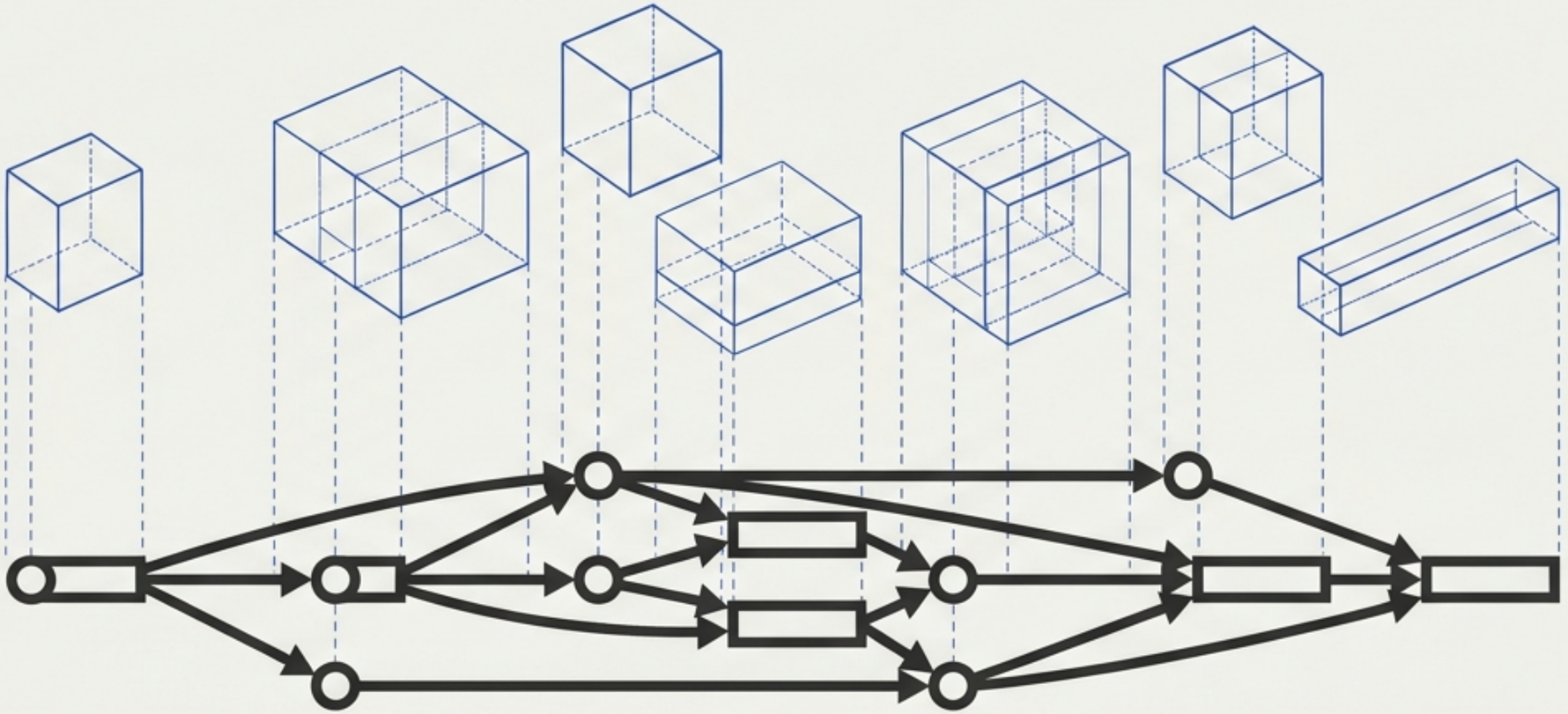
# History-Native Runtimes: The Projected Multiverse

## Ontology, Admissibility, and the Preservation of Refused Futures



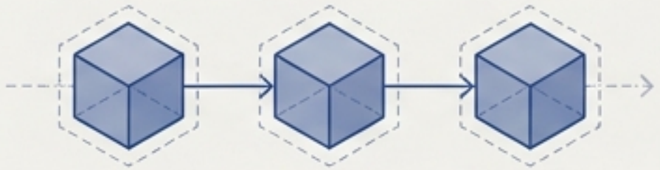
“Every computational system implicitly answers one question: What is the fundamental object?”

“Modern systems give different answers, but they all share a blind spot. They determine what a system can ask about itself by prioritizing the present over the past.”

In all three, State is primary. History is an optional annotation.

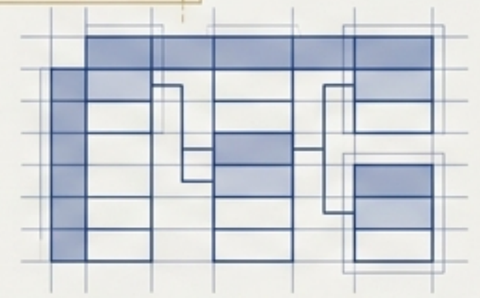
Unix: Mutable File System

Answers: The current state. Cannot recover the sequence of writes.



Git: Immutable Snapshot

Answers: The content hash. Cannot recover the keystrokes or semantics.



RDBMS: Fact Table

Answers: Data valid at query time. Cannot report the update history without explicit audit tables.

# Moving from Byte-Level Tracking to Ontological Memory

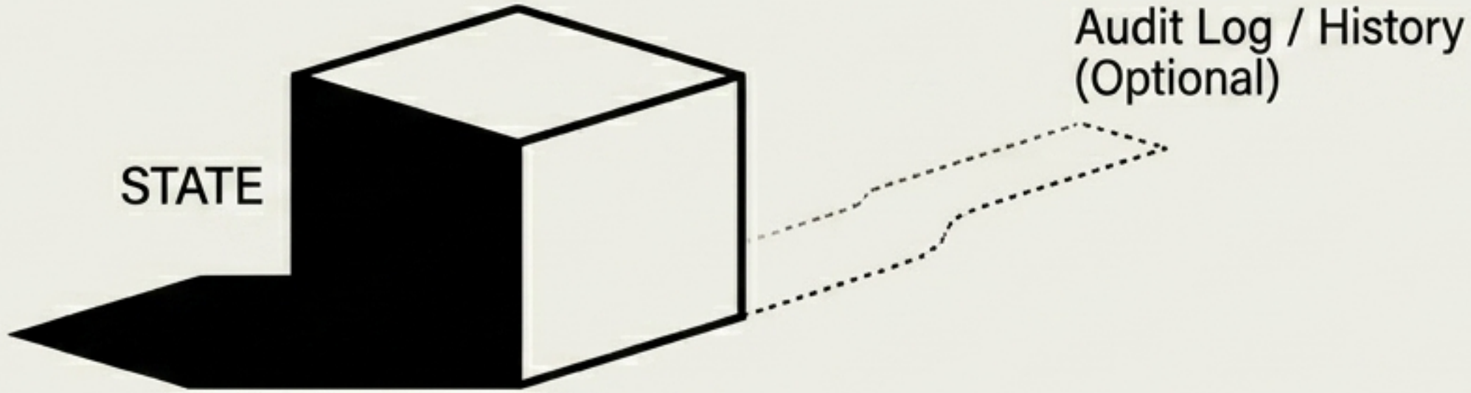
Event sourcing and provenance systems edge closer to history as a primitive, but only History-Native Runtimes track informational semantics and failed futures.

| | Unix | Git | Provenance | Event Sourcing | **History-Native** |
|---|---|---|---|---|---|
| Fundamental Object | State | Snapshot | Provenance Graph | Event Sequence | **History** |
| Integrity Model | None | Content-Addressed Hash | Varies | Application-defined | **Content-Addressed DAG** |
| Tracks Semantic Distinctions | No | No | Partially | No | **Yes (Ledger)** |
| Exposes State Fibers | No | No | No | No | **Yes** |
| Stores Refusals (Blocked Futures) | No | No | No | No | **Yes (First-Class)** |

# The Ontological Inversion: State is a Hologram

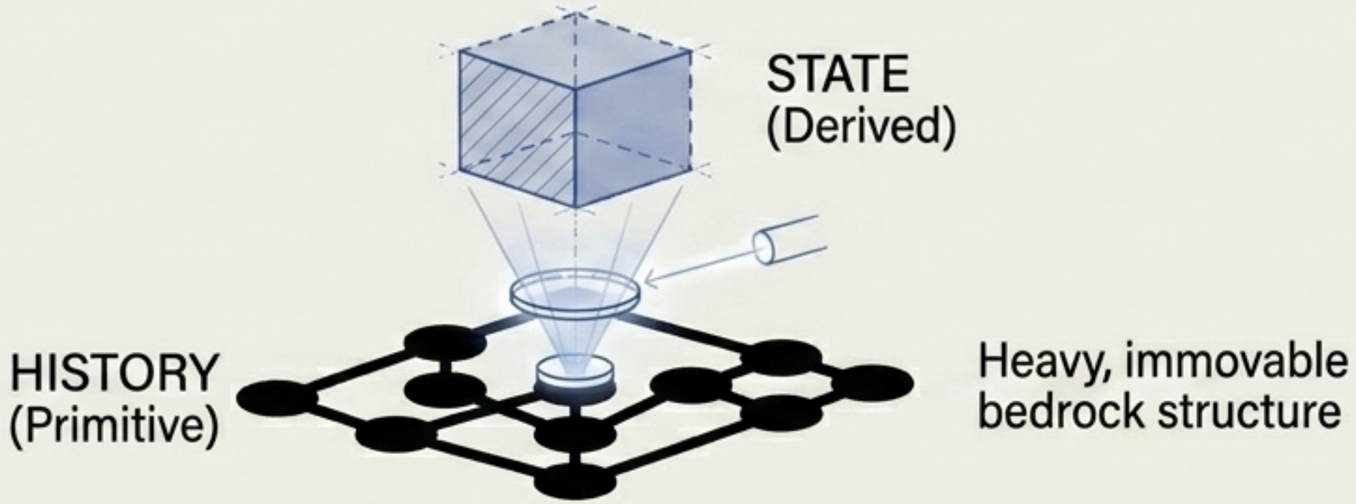
In a history-native architecture, histories are ontologically primitive. States are derived objects, reconstructed on demand rather than stored.

Conventional:
State-First



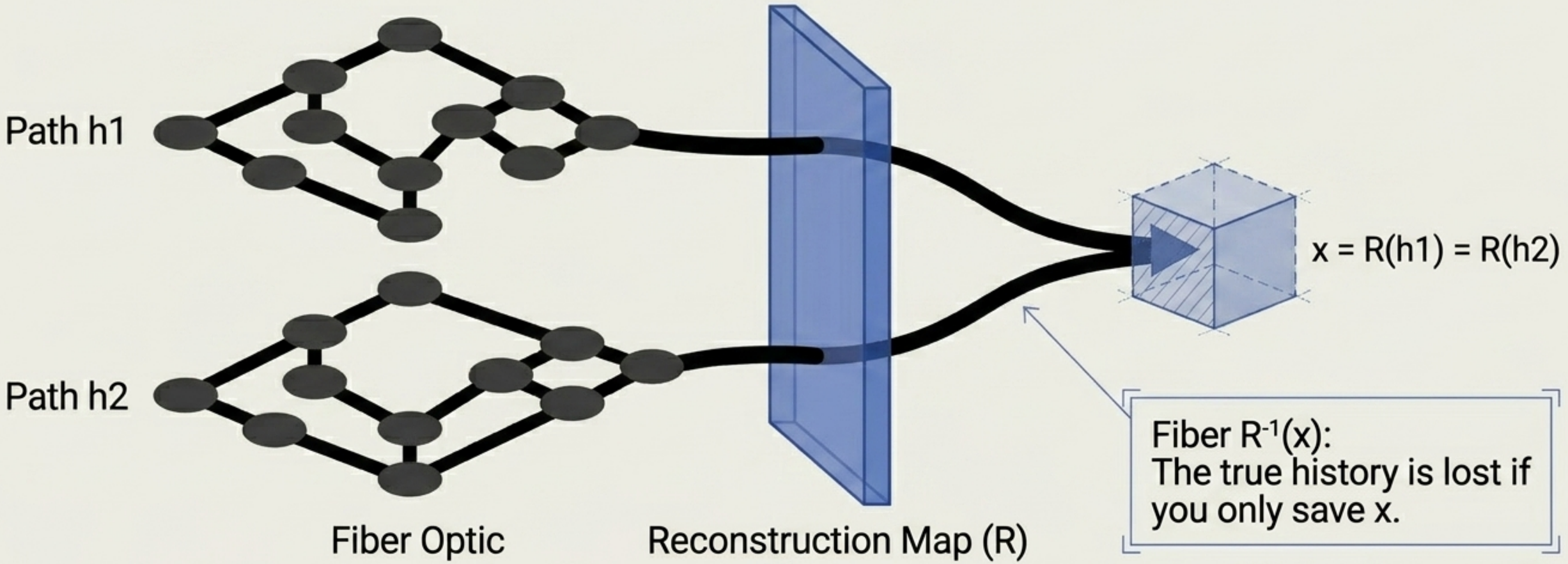

---

History-Native:
History-First



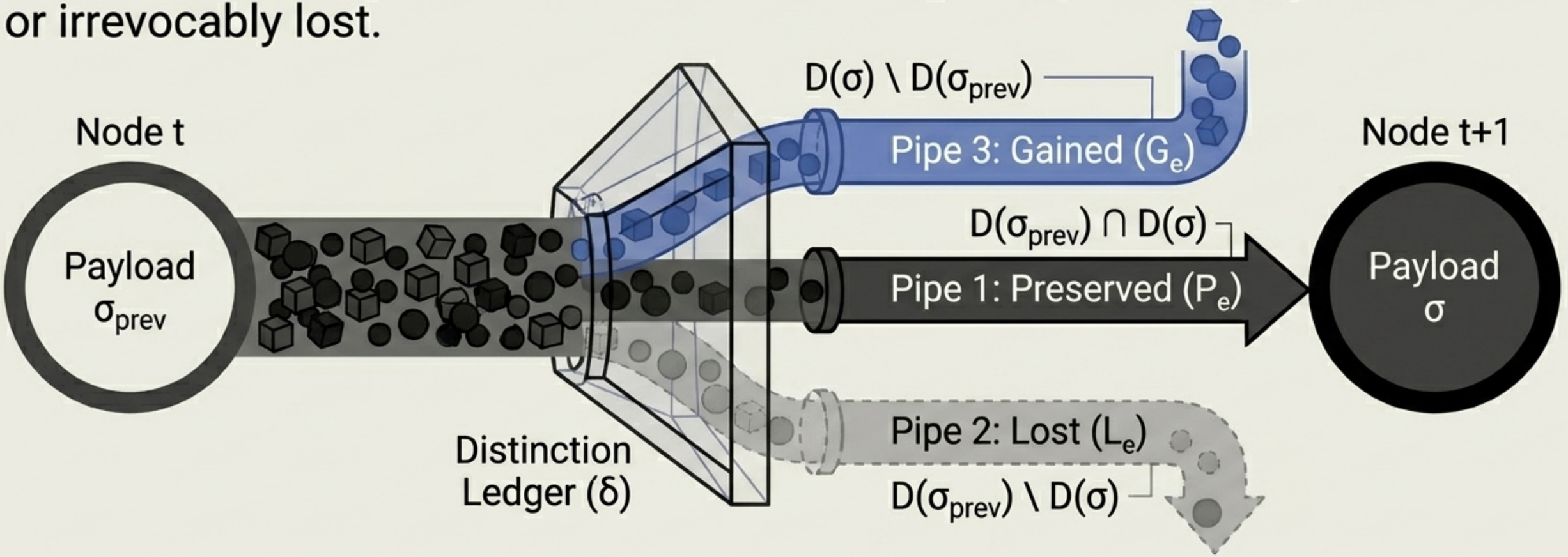
# The Reconstruction Map & The Invisible Fiber

Because state is a derived projection (R: H → X), distinct histories can produce the exact same state. This equivalence class is called a fiber. To state-centric systems, this structure is completely invisible.



# The Distinction Ledger: Tracking Informational Boundaries

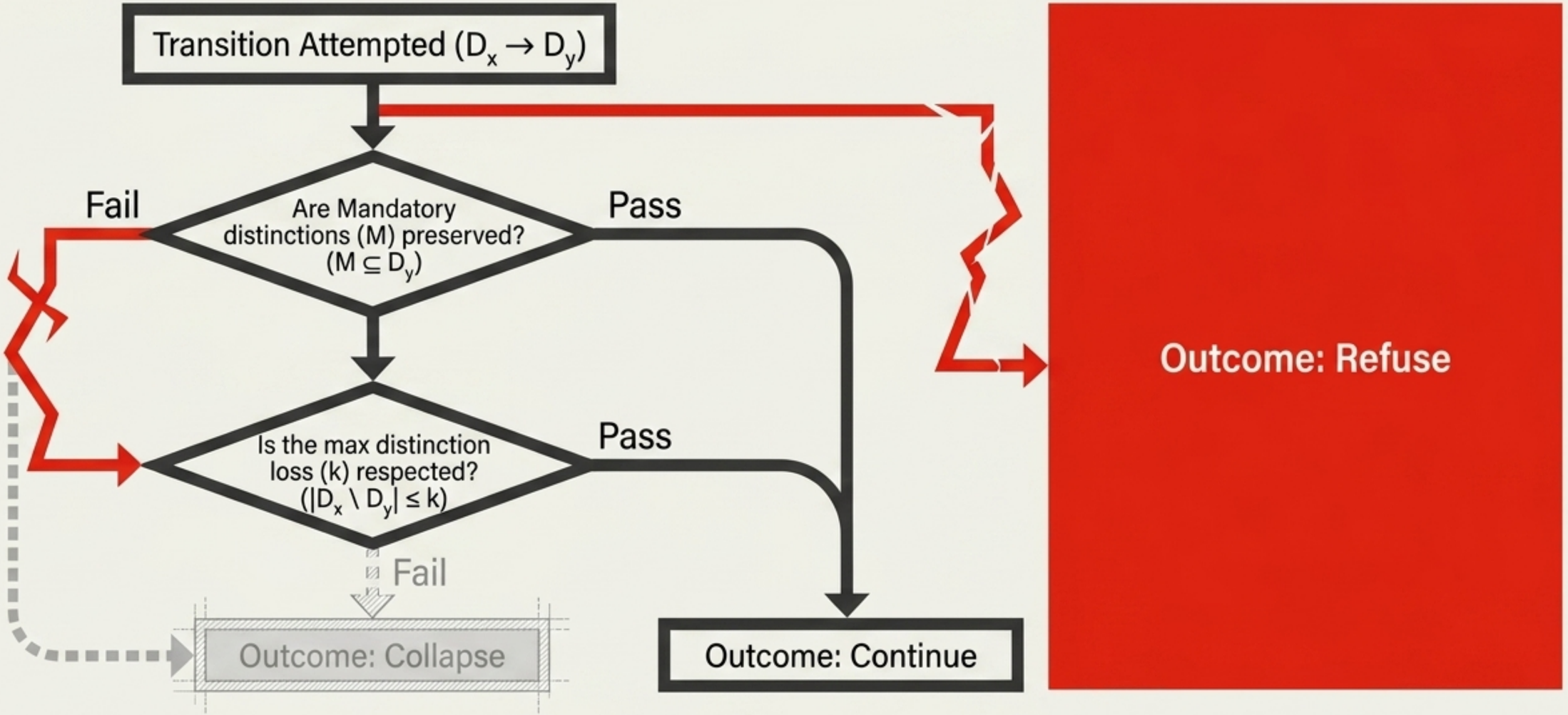
Instead of byte-level diffs, events record semantic boundaries. At each transition, the ledger calculates what meaning was preserved, gained, or irrevocably lost.




The Verifier Rule: $D_{t+1} = (D_t \setminus L_t) \cup G_t$

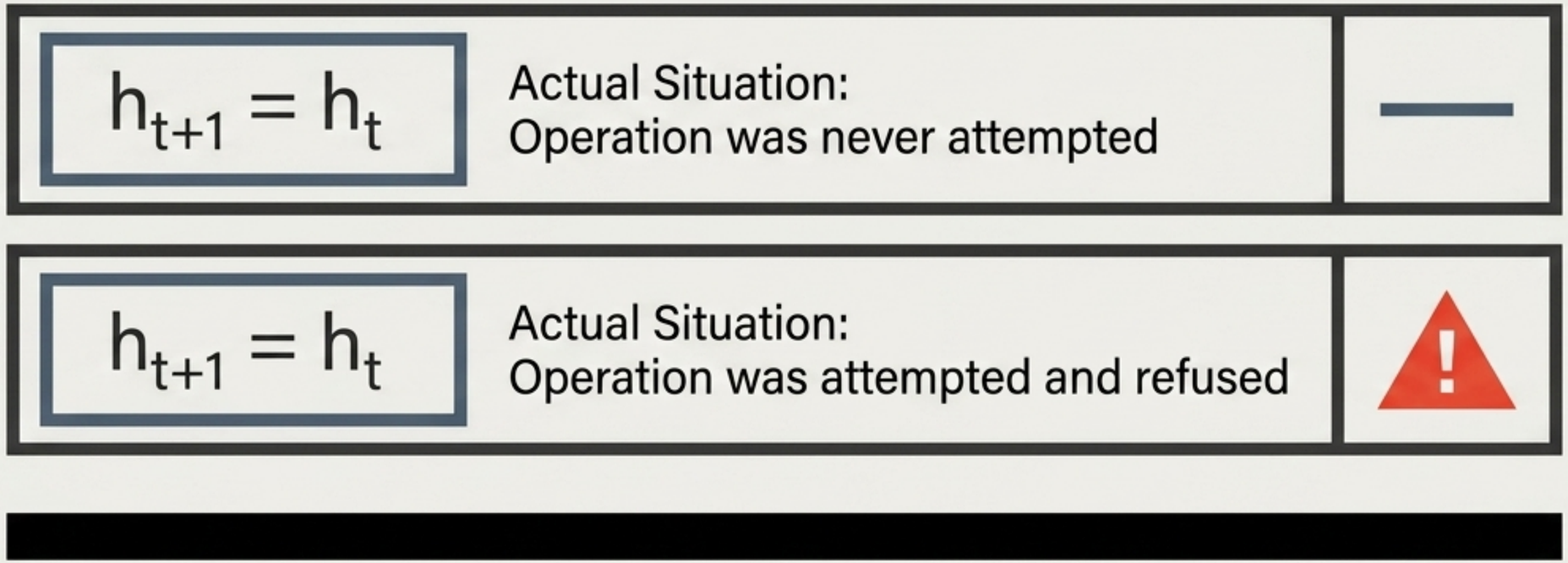
# Admissibility Policies: The Physics of the Runtime

Operations are governed by policies $\Pi = (M, k)$ that evaluate distinction loss in real-time. Failing the policy does not trigger an error code—it triggers an architectural divergence.



# The Informational Cost of Conventional Design

When a policy blocks an action, conventional systems erase the attempt. Downstream, it is impossible to distinguish between a system at rest and a system actively suppressing a forbidden state.



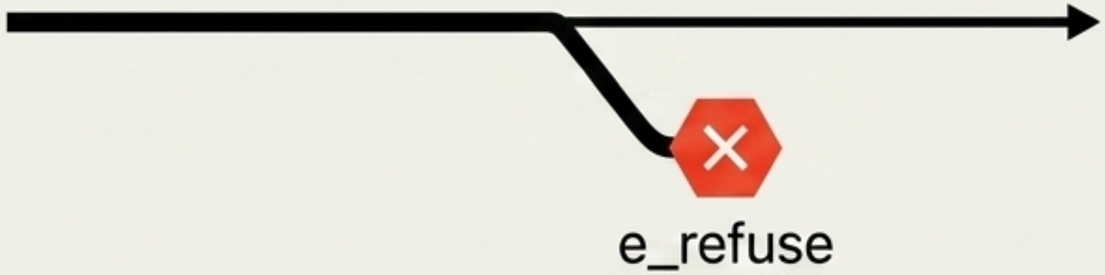

Because they look identically formulated, the true operational intent is destroyed.

# The Superpower: Preserving the Refused Future

A history-native runtime does not discard rejected transitions. It crystallizes them as first-class history nodes, proving exactly what was tried, why it was stopped, and what distinctions would have been lost.

Conventional

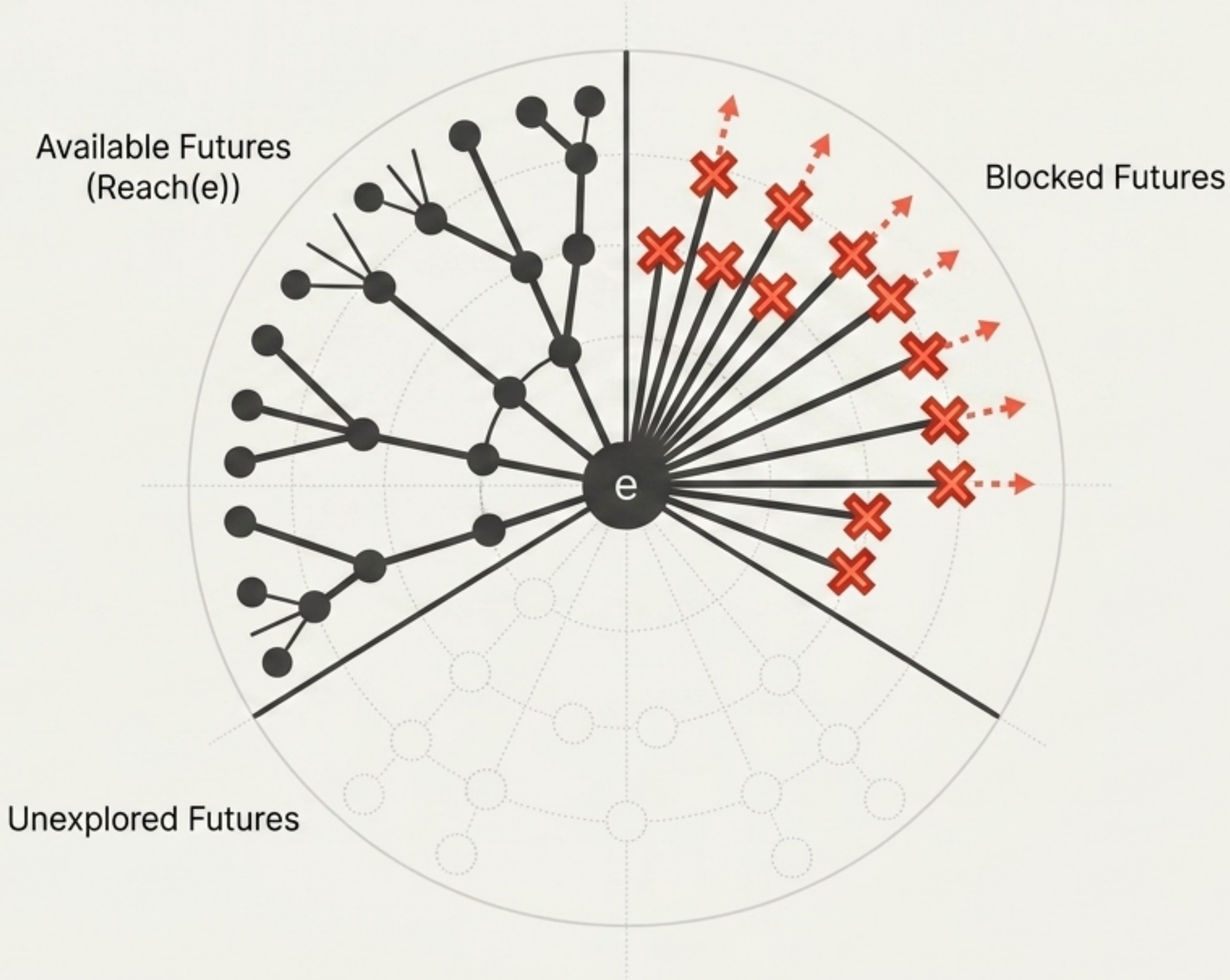
History-Native




$$h_{t+1}^{\mathrm{HN}} = h_t \parallel e_{\mathrm{refuse}}$$

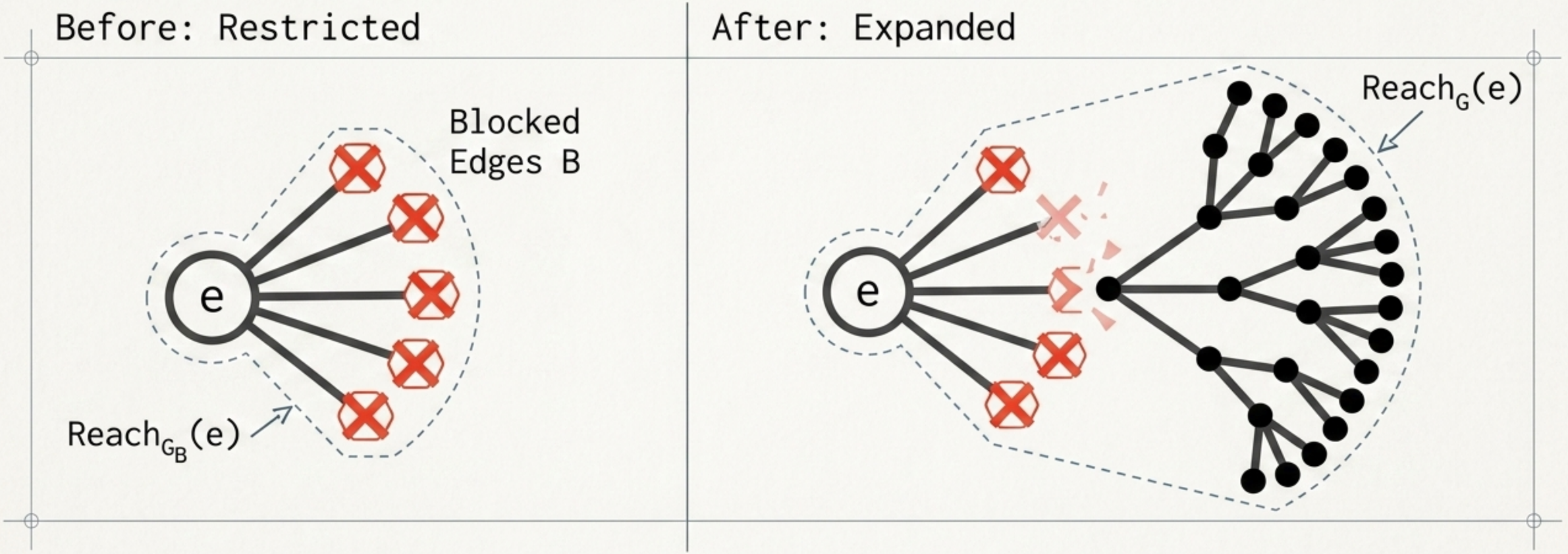
# Reachability: Mapping the Multiverse of Futures

By preserving refusals, the system distinguishes between futures we haven't explored yet, and futures the system actively prevents us from reaching.



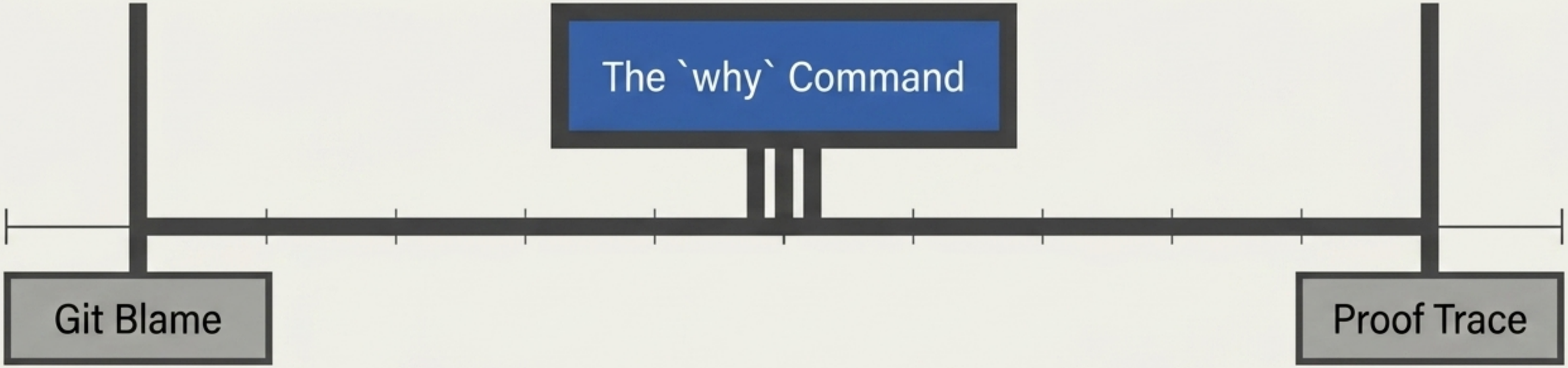
# The Repair Expansion Theorem

Repair in a history-native runtime does not mean editing a mutable state. It means removing blocked edges (refusal policies) to physically expand the reachable future set. Repair never decreases reachability.



# Causal Explanation: The `why` Command

A completely complete, distinction-faithful explanation. It reconstructs the causal ledger to show not just who changed a system, but what semantics survived the journey.




Focus: Authorship & Bytes

Who wrote this line?

Focus: Causality & Informational Semantics

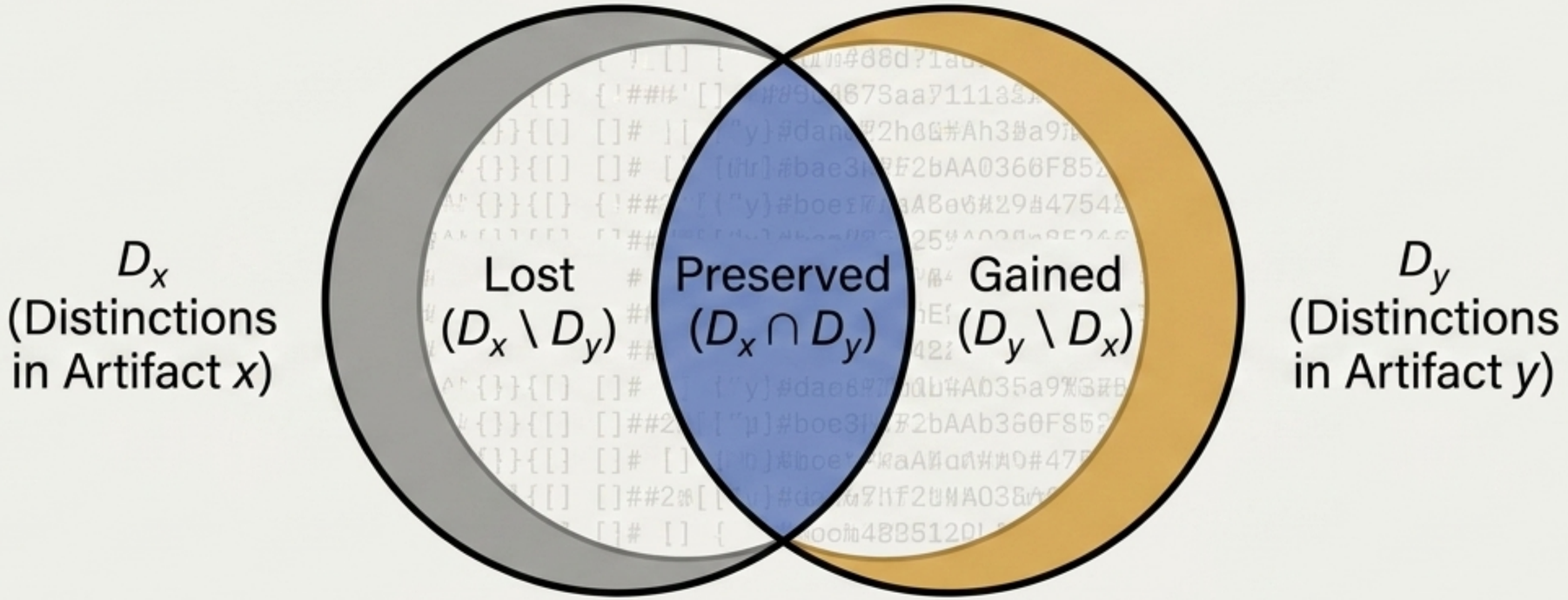
**What distinctions were preserved, lost, or refused on the path from origin to current state?**

Focus: Formal Rules

How was this derived?

# Semantic Diff: The Anatomy of Change

Two payloads may be byte-for-byte different yet semantically identical. A semantic diff performs a strict set decomposition over the distinction space, proving exactly what structural meaning shifted.




These sets form a perfect, pairwise disjoint partition of $D_x \cup D_y$.

# Executable Reality: The `cprsh` Prototype

The ontological claims have computationally verifiable content. The prototype instantiates the formal DAG as a content-addressed JSON append-only ledger.

```
{
   "op": "create",
   "hash": "<sha256 of payload>",
   "parents": [],
   "payload": "...",
   "ledger": {
      "preserved": [...], "lost": [...], "gained": [...],
      "loss": 0, "gain": 3,
      "count_before": 0, "count_after": 3 },
   "time": 1234567890.0
}
```

Merkle DAG Root (Lemma 2.8)

Distinction Tracker

Shell Commands to Formal Theorems Mapping

| Shell Commands | Formal Theorems |
|---|---|
| reach | Definition 5.1 |
| repair | Theorem 5.3 |
| fiber | Theorem 2.5 |
| why | Theorem 7.2 |
| sdiff | Theorem 8.1 |

# The Practical Consequences of Inversion

Shifting from “State-First” to “History-First” completely rewires how we **interrogate, fix,** and audit systems.

| Daily Workflow | State-Centric Paradigm (Old) | History-Native Paradigm (New) |
|---|---|---|
| Debugging | What is the current state? | Which event produced the current distinction structure? |
| Repair | Modify the mutable state. | Remove blocked edges to expand the reachable future. |
| Audit | Read the optional text log. | Cryptographically verify the causal ledger chain. |