Decision-Valued Maps: A Diagnostic Infrastructure for Representational Dependence

Anonymous

January 24, 2026

Abstract

Complex analytical pipelines produce discrete outcome identities that depend on representational choices in ways that are rarely recorded or examined. This paper formalizes decision-valued maps as the central diagnostic object for making such dependencies observable. The paper describes a minimal infrastructure for logging, replaying, and auditing the mapping from representation families to discrete decision identities under fixed snapshots and engines. The empirical method employs canonical representational sweeps that partition representation space into regions of identity persistence and boundary formation. The contribution is infrastructural: a system-level abstraction that renders representational dependence empirically testable without introducing performance targets, learning dynamics, or outcome-improvement claims.

1 Introduction

Complex analytical pipelines deployed in scientific, engineering, and policy settings routinely produce discrete outcome identities that appear stable under nominal conditions yet remain sensitive to subtle representational choices. Small variations in encoding, preprocessing rules, aggregation policies, or structural descriptions can induce qualitatively different outcomes, even when the underlying data snapshot and computational engine are unchanged. Because these dependencies are rarely recorded or examined as first-class objects of analysis, they are often discovered only after deployment, failure, or post-hoc audit.

Current practice tends to focus on optimizing performance with respect to fixed representations, while representational choices themselves are treated as incidental implementation details. As a result, the mapping between families of representations and the resulting discrete outcomes remains implicit. This limits the ability to assess persistence, detect boundary formation, or anticipate fracture points where outcome identity changes.

This paper introduces a diagnostic perspective centered on making such dependencies observable. The paper formalizes a decision-valued mapping from representations to discrete outcome identities and describes a minimal infrastructure for logging, replaying, and auditing this mapping under controlled variation. The contribution is not a new model, learning rule, or optimization procedure, but a system-level abstraction that renders representational dependence empirically testable.

2 Problem Setting and Scope

The framework considers systems that operate on a frozen snapshot of the world and produce discrete outcome identities via a fixed computational engine. A snapshot is a bounded, immutable

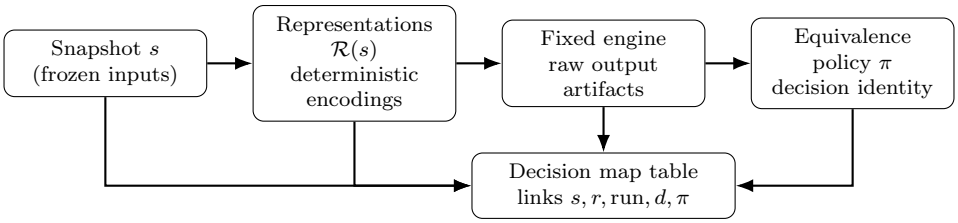

Figure 1: **Canonical decision-valued mapping schematic for a fixed snapshot and engine.** A frozen snapshot is deterministically encoded into multiple representations. Each representation is consumed by the same fixed engine, producing immutable raw outputs. Discrete decision identity is extracted by a declared equivalence policy. The decision map materializes the links among snapshot, representation, engine run, decision, and policy, making representational dependence empirically testable. Minimal example: fix s and engine, vary $r \in \mathcal{R}(s)$, extract $d \in \mathcal{D}$ using π.

slice of inputs over a specified time window. A representation is a deterministic encoding of that snapshot, defined by explicit structural choices such as kernels, thresholds, weighting rules, or aggregation policies. An engine is a fixed solver, simulator, or inference routine that consumes a representation and produces raw output. A decision is a discrete identity extracted from this output according to a declared equivalence policy.

The focus of this work is diagnostic rather than prescriptive. The framework does not introduce training procedures, adaptive updates, gradient-based optimization, or online learning. Each change to the world state is treated as a new snapshot, and each representational variation is explicitly declared. The goal is to characterize when outcome identities persist across representational variation and when they fracture, not to improve or optimize the outcomes themselves.

This scope intentionally excludes continuous outputs unless they are reduced to discrete identities via a declared policy. It also excludes claims about cognition, intelligence, or learning. The framework applies wherever discrete outcomes are produced by complex pipelines and where representational choices may influence those outcomes in non-obvious ways.

3 Core Object

The central object of study is a decision-valued map

$$f : \mathcal{R} \to \mathcal{D},$$

where $\mathcal{R}$ denotes a family of representations over a fixed snapshot and $\mathcal{D}$ denotes a set of discrete decision identities.

Each element $r \in \mathcal{R}$ is a fully specified, deterministic representation derived from the snapshot. The engine consumes r and produces raw output, which is then reduced to a decision identity $d \in \mathcal{D}$ according to an equivalence policy. The equivalence policy defines when two raw outputs are considered to represent the same decision identity, independent of incidental numerical differences.

The purpose of the system is to make the mapping f queryable, replayable, and auditable. By materializing this map across controlled variation in $\mathcal{R}$, it becomes possible to observe regions of identity persistence, identify boundary formation, and detect fractures where small representational changes induce discrete outcome changes.

4 Experimental Protocol for Representational Sweeps

The diagnostic protocol proceeds in five stages. First, a snapshot is frozen and assigned a content-addressed identifier. Second, a representation family is declared, along with a deterministic factory that generates individual representations from parameter settings. Third, a sweep plan specifies the set of representational variants to be evaluated. Fourth, the fixed engine is executed independently for each representation, producing raw outputs that are stored as immutable artifacts. Fifth, a decision extractor applies a declared equivalence policy to produce discrete decision identities.

All artifacts are versioned and linked through content-addressed identifiers. Representations are distinct from tuning parameters, and engine configuration is held fixed across the sweep. The resulting materialized map from representations to decision identities supports reproducible replay and post-hoc analysis without re-running the engine. The following section instantiates this protocol with a minimal concrete example.

5 Minimal Example

This section provides a minimal end-to-end instance of the object studied in this paper: a decision-valued map from representations to discrete decision identities under a fixed snapshot and a fixed engine. The goal is not to introduce an application domain, but to make the dependency chain explicit and checkable.

5.1 Snapshot, representation family, engine, policy

The example begins by freezing a snapshot s that contains all external inputs required to execute the engine. For a concrete but domain-agnostic example, s can be a finite directed graph with node set V, edge set E, and an immutable set of edge attributes (for example, baseline costs). The snapshot identifier is content-addressed, so any byte-identical snapshot yields the same snapshot id.

A representation family $\mathcal{R}(s)$ is then declared as a deterministic set of encodings of s. In this minimal example, a representation includes two components: (i) an explicit query spec (for example, `start_node` and `end_node`) and (ii) a structural rule that deterministically transforms edge attributes into the engine's cost surface (for example, an aggregation rule or kernel choice). Each representation $r \in \mathcal{R}(s)$ is fully specified and content-addressed.

The engine is fixed. It consumes r and produces a raw output artifact. The engine configuration and version are held constant across representational variation. Raw outputs are written once and stored as immutable artifacts linked by content hash.

Finally, an equivalence policy π is declared that reduces raw engine output to a discrete decision identity. The policy defines when two raw outputs correspond to the same identity, independent of incidental numerical differences. This step turns a raw artifact into an element of $\mathcal{D}$ and makes the mapping $f : \mathcal{R} \to \mathcal{D}$ empirically materializable.

5.2 Two representations that differ only by structure

Fix a single snapshot s and a single query (the same `start_node` and `end_node`). Consider two representations r_0 and r_1 that differ only in the structural rule used to construct the cost surface from the frozen edge attributes of s. For example, r_0 may encode costs via one deterministic aggregation rule and r_1 via another deterministic aggregation rule, while all other declared fields remain identical.

The fixed engine is executed once per representation, producing two raw outputs, and then apply the same equivalence policy π to extract two decision identities d_0 and d_1. This produces two entries in the decision map table:

$$(r_0 \mapsto d_0), \qquad (r_1 \mapsto d_1).$$

If $d_0 = d_1$, the decision identity persists under this representational variation. If $d_0 \neq d_1$, the map exhibits a boundary between these two representations.

5.3 Replay verification as the minimal audit

The minimal audit claim supported by this example is replay determinism: re-running the same pipeline under the same snapshot id, representation id, engine version, and policy id yields identical identifiers for the resulting decision. Concretely, replay verification checks that the same decision id and the same decision payload hash are recovered across independent runs, without writing new rows or changing bindings. This is the smallest unit of evidence that the mapping f is not merely described, but operationally testable.

5.4 From the minimal example to representational sweeps

The minimal example yields a map with two points in representation space. A representational sweep generalizes this by evaluating a declared set of representations spanning a parameterized family, then visualizing the resulting partition of representation space into regions of decision identity persistence separated by boundaries where identity changes. No continuous performance metric is required, because the empirical object is the structure of the decision-valued map itself.

6 Related Infrastructural Precedents

The perspective advanced here follows a recurring pattern in the development of technical infrastructure, in which latent dependencies within complex systems are rendered explicit, inspectable, and stable enough to support collective use. In software engineering, abstract data types formalized the separation between representation and observable behavior, allowing systems to evolve internally without collapsing external guarantees [2]. In database systems, write-ahead logging transformed durability and recovery from ad-hoc mechanisms into auditable, replayable state transitions [1]. In empirical research, specification-curve analysis made visible the dependence of reported conclusions on analytic choices that were previously implicit [3].

These precedents are not treated as peer systems or comparable products, but as examples of a shared infrastructural sensibility. In each case, the primary contribution was not a novel computational procedure or performance improvement, but the introduction of a diagnostic layer that made structural dependence observable and testable. Decision-valued mapping extends this tradition to settings where discrete outcomes emerge from complex analytical pipelines, and where representational choices exert nontrivial influence on outcome identity.

7 DecisionDB System Architecture

The core contribution of this work is a governance-aware diagnostic abstraction, not a new predictive method. Representational choice is treated as a first-class experimental variable, making the resulting dependence of discrete outcome identity on representation empirically testable. DecisionDB

operationalizes this by (i) content-addressing snapshots, representations, and outputs, (ii) holding the engine fixed while varying only declared representation structure, and (iii) defining decision identity through an explicit equivalence policy rather than through raw output. The resulting decision map is replayable and auditable, enabling direct observation of identity persistence, boundary formation, and fractures induced by representational variation, without introducing optimization objectives, learning dynamics, or performance claims.

DecisionDB implements a minimal diagnostic infrastructure for materializing and auditing decision-valued maps of the form

$$f : \mathcal{R} \to \mathcal{D},$$

where representations vary under a fixed snapshot and a fixed engine. The system is designed to make representational dependence observable without introducing new models, performance targets, or adaptive procedures.

7.1 Identifiers and Content Addressing

All core objects in DecisionDB are identified using content-addressed hashes computed over canonical JSON encodings. Canonicalization rules enforce deterministic serialization by sorting keys, preserving array order, excluding whitespace, serializing floats as strings, and including explicit version fields. Hashes are computed using SHA-256, and identifiers are formed by prefixing the first sixteen hexadecimal characters of the digest with a type-specific tag.

This scheme ensures that identical content always produces identical identifiers, enabling reproducible replay and audit across runs and environments.

7.2 Core Entities

DecisionDB tracks five primary entities.

Snapshots represent frozen slices of the world over a declared time window. A snapshot captures all external inputs required for downstream processing and is treated as immutable once created.

Representations are deterministic encodings of a snapshot. Each representation is generated by a declared factory and fully specified by its representation specification, namespace, and factory version. Representation parameters are explicitly separated from tuning or engine parameters.

Engine runs record executions of a fixed computational engine on a specific representation. Engine configuration and version are held constant across representational sweeps. Raw outputs are stored as immutable artifacts and linked by content hash.

Decisions are discrete outcome identities extracted from raw engine output using a declared equivalence policy. The equivalence policy defines when two outputs correspond to the same identity, independent of incidental numerical differences.

Decision maps materialize the mapping between representations, engine runs, and decision identities. This table constitutes the empirical object of analysis and supports queries over persistence, boundary formation, and fracture.

7.3 Protocol Enforcement

DecisionDB enforces four invariants. First, reproducibility requires identical inputs to yield identical identifiers. Second, auditability requires that every mapping link to versioned artifacts and policies. Third, separation ensures that representation parameters are not conflated with engine tuning. Fourth, identity stability requires that decision identity be defined by policy rather than raw output.

Figure 3: System Architecture
<ARCHITECTURE_DIAGRAM>

Figure 2: System architecture for logging and replaying decision-valued maps. Content-addressed identifiers link snapshots, representations, engine runs, and decisions to ensure reproducibility and auditability.

By enforcing these constraints at the infrastructure level, DecisionDB supports empirical analysis of representational dependence without requiring reinterpretation of system behavior or rerunning engines post hoc.

8 Empirical Characterization via Representational Sweeps

Empirical analysis proceeds by materializing decision-valued maps across controlled representational sweeps. For a fixed snapshot and engine, a declared representation family defines a finite or discretized set of representations. Each representation is evaluated independently, and a decision identity is extracted using a fixed equivalence policy.

The resulting map partitions representation space into regions of identity persistence, separated by boundaries where decision identity changes. Persistence regions indicate ranges of representational variation over which outcome identity remains stable. Boundaries mark loci of sensitivity where small representational changes induce discrete outcome changes. Fractures correspond to abrupt transitions in identity that cannot be explained by gradual variation in representation parameters.

Analysis focuses on describing the geometry and topology of these regions rather than optimizing outcomes. No performance metrics, loss functions, or preference orderings are introduced. The empirical object is the structure of the decision-valued map itself.

By comparing sweeps across different representation families applied to the same snapshot and engine, it becomes possible to distinguish representation-induced variability from changes attributable to the underlying world state. This supports diagnostic assessment of robustness, auditability, and failure modes in complex analytical pipelines.

8.1 Canonical Sweep Visualization

Figure 2: Primary Representational Sweep
<EXP_ID_PRIMARY>

Figure 3: Example representational sweep showing regions of decision identity persistence and boundary formation. Each point corresponds to a representation variant; color indicates discrete decision identity.

All empirical results in this work are expressed through a single canonical visualization of a

representational sweep. The purpose of this visualization is not to summarize performance or optimize outcomes, but to make the structure of the decision-valued map observable.

Each sweep visualization corresponds to a fixed snapshot, a fixed engine, and a declared family of representations. The axes of the visualization are defined by explicit representation parameters drawn from the representation specification. Each evaluated representation occupies a single point in this parameter space.

Decision identity is encoded categorically, for example by color or region label. No continuous performance metric is displayed. The visualization is constructed such that identical decision identities are visually grouped, making regions of persistence immediately apparent. Boundaries correspond to loci where decision identity changes as representation parameters vary. Fractures correspond to abrupt identity changes induced by small representational variation.

All sweep visualizations conform to this structure. Differences between figures reflect differences in snapshots, engines, representation families, or equivalence policies, not changes in visualization semantics. This constraint ensures that figures remain comparable across experiments, manuscripts, proposals, and presentations.

9 Failure Modes and Misinterpretations

This section delineates the boundaries of applicability of the diagnostic framework and clarifies what claims are not made.

9.1 Discrete Outcome Requirement

DecisionDB applies only to systems where outputs can be reduced to discrete identities via a declared equivalence policy. Continuous outputs are out of scope unless such a reduction is explicitly defined.

9.2 Fixed Snapshot and Engine Assumption

All empirical results assume a frozen snapshot and a fixed engine. Changes to data, model parameters, or execution logic constitute a new analytical context and require a new analysis.

9.3 No Claims About Optimization or Learning

This framework does not improve outcomes, select better representations, or introduce learning dynamics. Any apparent performance effects observed in sweeps are incidental and not interpreted as optimization results.

9.4 Empirical Coverage Limits

Empirical Result Placeholder:

- Representation families evaluated: `<REP_FAMILY_LIST>`
- Regions not evaluated: `<UNEXPLORED_REGIONS>`

Reported sweeps cover only declared representation families and parameter ranges. Unobserved regions of representation space remain unconstrained.

10 Discussion

This section interprets the empirical structure of decision-valued maps produced by representational sweeps in terms of auditability, reproducibility, and system reliability. The emphasis is on diagnosing failure modes and structural sensitivity rather than improving outcomes or performance.

10.1 Auditability via Explicit Decision Provenance

Figure 5: Reproducibility Replay Verification
<EXP_ID_REPLAY>

Figure 4: Reproducibility replay verification. Independent rerun confirming identical decision identifiers are recovered across executions.

Empirical Result Placeholder:

- Experiment ID: <EXP_ID_AUDIT>
- Snapshot: <SNAPSHOT_ID>
- Representation family: <REP_FAMILY>
- Decision type: <DECISION_TYPE>

This subsection will report how decision identities can be traced deterministically to representation specifications, engine versions, and equivalence policies. The empirical contribution will consist of a verified replay in which identical identifiers are recovered across independent executions, demonstrating end-to-end auditability.

10.2 Reliability as Identity Persistence Under Variation

Empirical Result Placeholder:

- Experiment ID: <EXP_ID_PERSISTENCE>
- Representation parameters swept: <PARAM_LIST>

This subsection will characterize regions of representation space over which decision identity remains unchanged. Persistence will be reported descriptively, without ranking or optimization, as a property of the materialized decision-valued map.

10.3 Failure Precursors and Boundary Localization

Empirical Result Placeholder:

- Experiment ID: <EXP_ID_BOUNDARY>
- Observed boundary parameters: <BOUNDARY_PARAM_RANGE>

This subsection will document loci in representation space where decision identity changes. These boundaries will be interpreted as potential failure precursors that can be diagnosed prior to deployment or incident occurrence.

Figure 4: Boundary and Fracture Localization
<EXP_ID_BOUNDARY>

Figure 5: Boundary and fracture localization derived from representational sweep. Annotated parameter values indicate loci where decision identity changes.

11 Conclusion

This paper introduced decision-valued maps as a diagnostic object for characterizing how discrete outcome identities depend on representational choices. The infrastructure described here supports logging, replaying, and auditing these dependencies without introducing optimization, learning, or performance claims. Making representational dependence observable is a prerequisite for reliable reasoning about discrete outcomes in complex systems.

Glossary

Snapshot: A frozen, immutable slice of the world over a declared time window. Changes to the world state produce a new snapshot.

Representation: A deterministic encoding of a snapshot defined by explicit structural choices such as kernels, thresholds, weighting rules, or aggregation policies.

Engine: A fixed computational procedure that consumes a representation and produces raw output. Engine configuration and version are held constant during analysis.

Decision Identity: A discrete outcome extracted from engine output according to a declared equivalence policy, independent of incidental numerical variation.

Equivalence Policy: A declared rule that defines when two raw outputs correspond to the same decision identity.

Decision-Valued Map: The materialized mapping from representations to decision identities under a fixed snapshot and engine.

Representational Sweep: A controlled evaluation of a family of representations over a fixed snapshot and engine.

Persistence: A region of representational variation over which decision identity remains unchanged.

Boundary: A locus in representation space where decision identity changes.

Fracture: A discrete transition in decision identity induced by small representational variation.

References

[1] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, 1993.

[2] Barbara Liskov and John Guttag. *Programming with Abstract Data Types.* Addison-Wesley, 1978.

[3] Uri Simonsohn, Joseph Simmons, and Leif Nelson. Specification curve analysis. *PsyArXiv*, 2018.