Decision-Valued Maps: A Diagnostic Infrastructure for Representational Dependence

Gil Raitses

Abstract

Complex analytical pipelines produce discrete outcomes whose dependence on representational choices is rarely recorded or tested. We formalize decision-valued maps as a diagnostic object that makes this dependence observable: a mapping $f: \mathcal{R} \to \mathcal{D}$ from a family of representations to discrete decision identities, evaluated under a fixed data snapshot and a fixed computational engine. We describe DecisionDB, a minimal infrastructure for logging, replaying, and auditing such maps using content-addressed identifiers, immutable artifacts, and declared equivalence policies. In a concrete routing domain, we sweep two representation parameters across a fixed graph snapshot and fixed shortest-path engine. One parameter (neighbor weight) preserves route identity across its tested range; the other (second-order weight) induces a boundary where route identity changes. Independent replay verification confirms that all persisted decision identifiers are deterministically recoverable. The contribution is infrastructural: a system-level abstraction that partitions representation space into persistence regions and boundaries without introducing performance targets, learning dynamics, or outcome-improvement claims.

1 Introduction

Analytical pipelines in scientific, engineering, and policy settings routinely produce discrete outcomes: a selected route, a diagnostic label, a risk classification, a policy recommendation. These outcomes depend not only on the input data and the computational procedure, but also on representational choices: how the data is encoded, what features are weighted, which aggregation rules are applied. Small changes to such choices can induce qualitatively different outcomes, even when the underlying data and computation are unchanged.

These dependencies are real and consequential, yet they are rarely recorded as first-class objects of study. In standard practice, a pipeline is run under one representation, a result is reported, and the sensitivity of that result to representational alternatives goes unexamined. When instabilities surface, they are typically discovered after deployment or during post-hoc audits, because no infrastructure existed to make them visible earlier.

This paper introduces a diagnostic framework for making representational dependence observable. We define a *decision-valued map*, a mapping from a family of representations to discrete decision identities, evaluated under a fixed data snapshot and a fixed computational engine. By materializing this map across controlled representational variation, we can directly observe which outcomes persist under representation changes and where boundaries form.

We describe DecisionDB, a system that implements this framework. DecisionDB logs snapshots, representations, engine runs, and decision identities using content-addressed identifiers and immutable artifacts. It supports representational sweeps (systematic variation of declared representation parameters), replay verification (deterministic recovery of decision identifiers from persisted artifacts), and post-hoc audit of the full provenance chain.

We demonstrate the framework on a graph routing problem. Fixing a graph snapshot and a shortest-path engine, we sweep two representation parameters that control edge-cost construction. One parameter preserves route identity across its tested range; the other induces a discrete change in route identity at a specific threshold. Replay verification confirms that all persisted identifiers are deterministically recoverable.

The contribution is not a model, a learning rule, or an optimization procedure. It is an infrastructure for observing what happens to discrete outcomes when representations change.

2 Problem Scope

We consider systems with the following structure:

1. A **snapshot** s: a frozen, immutable slice of external inputs over a declared time window. Any change to the world state produces a new snapshot.
2. A **representation** $r \in \mathcal{R}(s)$: a deterministic encoding of s, defined by explicit structural choices (kernels, thresholds, weighting rules, aggregation policies). Each representation is fully specified by a declared parameter set and generated by a versioned factory.
3. A **engine** E: a fixed computational procedure that consumes a representation and produces raw output. Engine configuration and version are held constant during analysis.
4. An **equivalence policy** π: a declared rule that reduces raw engine output to a discrete **decision identity** $d \in \mathcal{D}$. The policy defines when two raw outputs correspond to the same identity, independent of incidental numerical differences.

The scope is diagnostic: we characterize when decision identity persists across representational variation and when it changes. We do not introduce training procedures, adaptive updates, gradient-based optimization, or online learning. Continuous outputs are in scope only when reduced to discrete identities via a declared policy.

This framing applies wherever discrete outcomes emerge from complex pipelines and representational choices may influence those outcomes. Examples include routing under alternative cost encodings, classification under alternative feature constructions, and resource allocation under alternative aggregation rules.

3 The Decision-Valued Map

The central object of study is a mapping

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

where $\mathcal{R}$ denotes a family of representations over a fixed snapshot s and $\mathcal{D}$ denotes a set of discrete decision identities. For each representation $r \in \mathcal{R}$, the engine E produces raw output $E(r)$, and the equivalence policy π extracts a decision identity $d = \pi(E(r))$.

Three structural features of this map are observable through controlled variation of $\mathcal{R}$:

Persistence regions. Connected subsets of $\mathcal{R}$ over which f is constant. Within a persistence region, representational variation does not change the outcome.

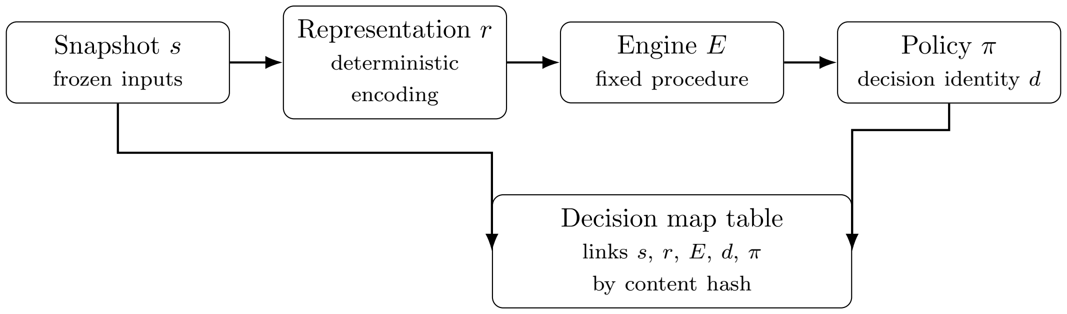


Figure 1: Decision-valued mapping pipeline. A frozen snapshot is encoded into a representation, consumed by a fixed engine, and reduced to a discrete decision identity via an equivalence policy. The decision map table materializes these links using content-addressed identifiers.

Boundaries. Loci in $\mathcal{R}$ where f changes value. A boundary separates two persistence regions with different decision identities.

Fractures. Boundaries where a small change in representation parameters induces a discrete identity change. Fractures indicate high sensitivity of the outcome to the representation.

The purpose of DecisionDB is to *materialize* f: to evaluate it at declared points in $\mathcal{R}$, store the results as immutable artifacts, and make the resulting map queryable, replayable, and auditable.

4 Sweep Protocol

A *representational sweep* evaluates the decision-valued map f across a declared set of representations. The protocol proceeds in five stages:

1. **Freeze snapshot.** A snapshot s is frozen and assigned a content-addressed identifier computed from its canonical JSON serialization (SHA-256, truncated to 16 hex characters, prefixed with `snap_`).

2. **Declare representations.** A representation family $\mathcal{R}(s)$ is declared, along with a deterministic factory that generates individual representations from parameter settings. Each representation receives a content-addressed identifier (prefix `repr_`). Representation parameters are explicitly separated from engine configuration.

3. **Plan sweep.** A sweep plan specifies the parameter grid to be evaluated, the engine name and version, and the equivalence policy. The plan itself is content-addressed.

4. **Execute engine.** The fixed engine is executed independently for each representation. Each run produces a raw output artifact stored as an immutable file, linked by content hash (prefix `run_`). Engine configuration is held constant across the sweep.

5. **Extract decisions.** The equivalence policy π is applied to each raw output to produce a discrete decision identity (prefix `dec_`). The decision map table records the link from representation to engine run to decision identity.

All artifacts are versioned and linked through content-addressed identifiers. The resulting materialized map supports reproducible replay and post-hoc analysis without re-executing the engine.

5 System Design

DecisionDB is implemented as a Python package backed by SQLite. It manages five entity types through a relational schema with content-addressed primary keys and foreign-key constraints.

5.1 Content Addressing

All identifiers are computed deterministically from content. Given an entity's payload (a Python dictionary), DecisionDB serializes it to canonical JSON (keys sorted alphabetically, no whitespace, arrays in declaration order, floats as strings, explicit version fields), computes the SHA-256 digest of the UTF-8 encoding, truncates to the first 16 hexadecimal characters, and prepends a type-specific prefix (`snap_`, `repr_`, `run_`, `dec_`, `pol_`, `exp_`). Identical content always produces identical identifiers, regardless of when or where the computation occurs.

5.2 Schema

The five core tables are:

`snapshots` stores frozen input state: snapshot identifier, time window, provenance metadata, and an artifact manifest listing all input files by content hash.

`representations` stores deterministic encodings of snapshots: representation identifier, a foreign key to the parent snapshot, the representation specification (as canonical JSON), factory version, and the artifact URI and hash of the generated encoding.

`engine_runs` stores execution records: run identifier, a foreign key to the representation consumed, engine name and version, configuration (as canonical JSON with its own hash), runtime in milliseconds, and the URI and hash of the raw output artifact.

`decisions` stores discrete outcomes: decision identifier, decision type, equivalence policy version, and the decision payload (as canonical JSON). The decision identifier is computed from the policy version and the payload, so the same raw output under different policies yields different decision identifiers.

`f_map` materializes the decision-valued map: a composite primary key of (representation identifier, engine run identifier), a foreign key to the decision, and optional auxiliary metrics. This table is the primary query surface for analyzing representation-to-decision relationships.

All writes are append-only, scoped by experiment identifier, and executed within transactions. Inserts use `INSERT OR IGNORE` for idempotency: re-inserting the same content-addressed entity is a no-op.

5.3 Equivalence Policies

An equivalence policy defines how raw engine output is reduced to a decision identity. Each policy specifies a *hash source* (which field of the raw output carries decision-relevant content, e.g., `route.nodes`), a *canonicalization rule* (how to serialize the extracted content, e.g., `json_sorted_keys_utf8`), and a *match rule* (how to determine identity, e.g., `sha256_equality`). The policy itself is content-addressed (prefix `pol_`), so any change to the policy definition produces a new policy identifier and new decision identifiers downstream.

5.4 Replay Verification

Replay verification checks that persisted decision identifiers are deterministically recoverable from stored artifacts. Given a persisted decision record, the verifier loads the raw engine output from its stored URI, recomputes the equivalence policy identifier from the stored policy specification, extracts the decision payload using the policy, recomputes the payload hash and decision identifier, and compares all recomputed values against the persisted values. Replay is read-only: it writes no new rows and modifies no existing state. A replay pass confirms end-to-end integrity of the content-addressing chain from raw output through policy application to decision identity.

6 Empirical Demonstration

We demonstrate the framework on a graph routing problem. The goal is not to evaluate routing algorithms, but to show how the decision-valued map makes representational dependence observable in a concrete setting.

6.1 Setup

Snapshot. We fix a directed graph with node set V ($|V| = 564$), edge set E, and immutable edge attributes including baseline costs derived from geographic distance and a normalized stress metric. The snapshot is content-addressed and frozen before any engine execution. We fix a single origin-destination query: start node 85, end node 50.

Representation family. Each representation encodes the graph's edge costs as a deterministic function of the frozen edge attributes. Two representation parameters control the cost surface: `neighbor_weight`, a weight applied to a neighbor-based cost component (tested at 0.5 and 1.0), and `second_order_weight`, a weight applied to a second-order cost component (tested at 0.25 and 0.5). Each sweep varies one parameter while holding the other fixed, producing two representation variants per sweep and four engine evaluations total.

Engine. A fixed shortest-path solver (Dijkstra's algorithm [4]), with configuration and version held constant across all runs. Execution times ranged from 0.5 to 1.4 milliseconds.

Equivalence policy. Policy `pol_d8da3e00e9584eb1` (version 1.0.0): exact match on the canonicalized node sequence of the computed route. Two routes are assigned the same decision identity if and only if they traverse identical node sequences. The policy uses SHA-256 over JSON-serialized, sorted-key, UTF-8-encoded route nodes.

6.2 Results: Identity Persistence and Boundary Formation

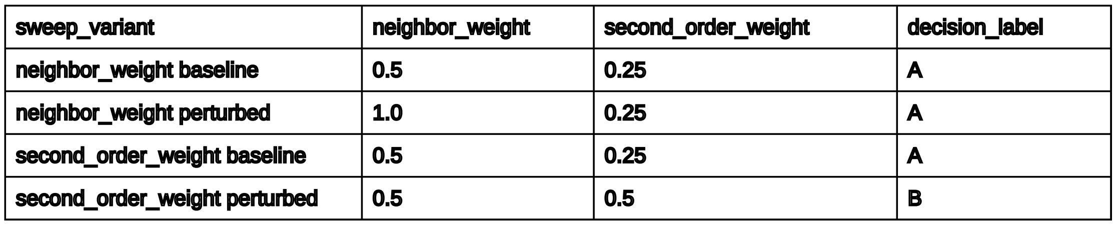
Neighbor weight sweep ($0.5 \to 1.0$, second-order weight fixed at 0.25). Both representations yield the same route:

Decision A: $[85, 176, 463, 14, 404, 76, 406, 407, 223, 200, 311, 310, 314, 322, 323, 50]$
(16 nodes, decision identifier `dec_e280...`).

Doubling the neighbor weight from 0.5 to 1.0 does not change route identity. This parameter range constitutes a persistence region under the tested policy.

Second-order weight sweep ($0.25 \to 0.5$, neighbor weight fixed at 0.5). The two representations yield different routes:

Decision A (second-order weight = 0.25): same route as above.
Decision B (second-order weight = 0.5): $[85, 411, 419, 422, 332, 204, 500, 369, 79, 402, 473, 502, 501, 50]$
(14 nodes, decision identifier `dec_c2dd...`).



sweep_variant	neighbor_weight	second_order_weight	decision_label
neighbor_weight baseline	0.5	0.25	A
neighbor_weight perturbed	1.0	0.25	A
second_order_weight baseline	0.5	0.25	A
second_order_weight perturbed	0.5	0.5	B

Figure 2: Primary representational sweep. Each column corresponds to a representation variant defined by its weight parameter setting. Decision identity (A or B) is assigned by the equivalence policy based on the route node sequence. The neighbor weight sweep (left) shows identity persistence; the second-order weight sweep (right) shows a boundary.

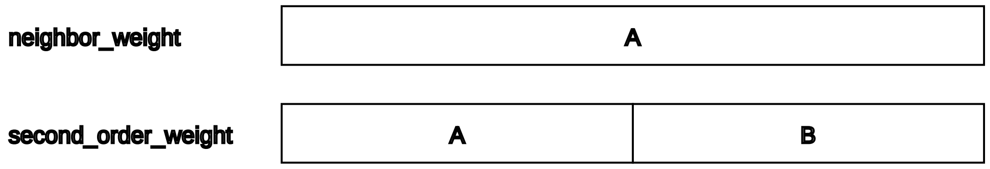


Figure 3: Identity persistence regions derived from the sweep in Figure 2. The neighbor weight parameter spans a single persistence region (Decision A). The second-order weight parameter spans two regions (A and B) separated by a boundary between 0.25 and 0.5.

The route changes entirely: a different sequence of 14 nodes instead of 16, traversing a different region of the graph. The boundary between these two decision identities lies between second-order weight values 0.25 and 0.5. This is a fracture: a small parameter change induces a qualitative change in the discrete outcome.

Table 1 summarizes the sweep results.

6.3 Replay Verification

We verify replay determinism for decision `dec_e28092c4dc33b8f1`, produced by engine run `run_36a0c96b084b2ffd`. The replay procedure loads the stored raw output (a 29-node route with 532 nodes explored), recomputes the equivalence policy identifier, extracts the decision payload, and recomputes both the payload hash and the decision identifier.

All recomputed values match the persisted values exactly:

The replay writes no new rows and modifies no database state. Table counts before and after replay are identical (1 engine run, 1 decision, 1 f_map entry). This confirms that the content-addressing chain from raw output through policy application to decision identity is deterministic and end-to-end auditable.

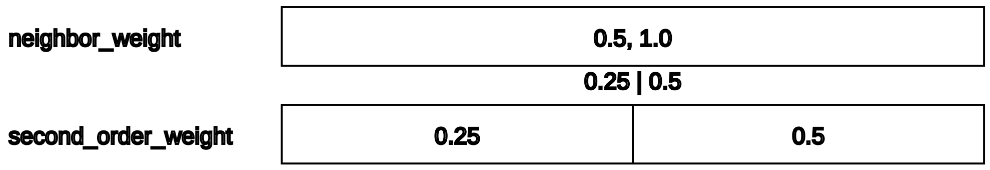


Figure 4: Boundary and fracture localization. The neighbor weight sweep shows no boundary (`route_changed = false`). The second-order weight sweep shows a boundary between 0.25 and 0.5 (`route_changed = true`, `edge_order_changed = true`).

Table 1: Summary of representational sweep results.

Sweep parameter	Value	Decision	Nodes	Boundary?
`neighbor_weight`	0.5	A	16	
`neighbor_weight`	1.0	A	16	No
`second_order_weight`	0.25	A	16	
`second_order_weight`	0.5	B	14	Yes

7 Related Work

The sensitivity of analytical conclusions to representational and analytic choices has been documented across multiple disciplines. This section situates the decision-valued mapping framework relative to existing work on analytic flexibility, reproducibility infrastructure, and provenance systems.

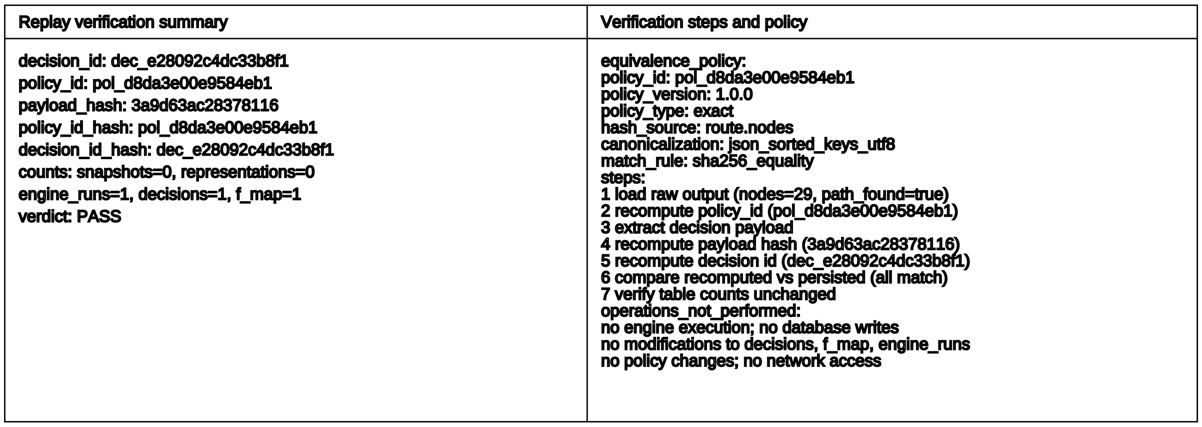
7.1 Analytic Flexibility and Multiverse Methods

Specification curve analysis [14] systematically evaluates how reported effects vary across defensible analytic specifications, making the dependence of statistical conclusions on analytic choices visible. Multiverse analysis [15] extends this idea by jointly varying data processing and model specification decisions to characterize the full space of results consistent with a dataset. The garden of forking paths [6] describes how implicit researcher degrees of freedom shape reported findings even absent deliberate p-hacking. The vibration of effects framework [11] quantifies how effect estimates fluctuate across model specifications in observational studies.

Decision-valued mapping shares the premise that analytic conclusions depend on choices that are often left implicit. The distinction is structural: specification curve and multiverse analyses operate on continuous effect estimates (e.g., regression coefficients, p-values) and visualize their distribution. Decision-valued maps operate on discrete outcome identities and characterize the topology of representation space, identifying which regions preserve identity and where boundaries form. The object of analysis is a partition, not a distribution.

Table 2: Replay verification results for decision `dec_e280...`.

Field	Persisted	Recomputed
Policy ID	`pol_d8da3e00e9584eb1`	`pol_d8da3e00e9584eb1`
Payload hash	`3a9d63ac28378116`	`3a9d63ac28378116`
Decision ID	`dec_e28092c4dc33b8f1`	`dec_e28092c4dc33b8f1`



Replay verification summary	Verification steps and policy
decision_id: dec_e28092c4dc33b8f1 policy_id: pol_d8da3e00e9584eb1 payload_hash: 3a9d63ac28378116 policy_id_hash: pol_d8da3e00e9584eb1 decision_id_hash: dec_e28092c4dc33b8f1 counts: snapshots=0, representations=0 engine_runs=1, decisions=1, f_map=1 verdict: PASS	equivalence_policy: policy_id: pol_d8da3e00e9584eb1 policy_version: 1.0.0 policy_type: exact hash_source: route.nodes canonicalization: json_sorted_keys_utf8 match_rule: sha256_equality steps: 1 load raw output (nodes=29, path_found=true) 2 recompute policy_id (pol_d8da3e00e9584eb1) 3 extract decision payload 4 recompute payload hash (3a9d63ac28378116) 5 recompute decision id (dec_e28092c4dc33b8f1) 6 compare recomputed vs persisted (all match) 7 verify table counts unchanged operations_not_performed: no engine execution; no database writes no modifications to decisions, f_map, engine_runs no policy changes; no network access

Figure 5: Replay verification log. Recomputed policy identifier, payload hash, and decision identifier match the persisted manifest. Database table counts remain unchanged, confirming read-only replay.

7.2 Sensitivity Analysis

Global sensitivity analysis [13] quantifies how variation in model inputs contributes to variation in model outputs, typically through variance decomposition or derivative-based indices over continuous output spaces. Decision-valued mapping differs in that it does not decompose output variance or compute sensitivity indices. Instead, it directly observes whether a discrete outcome changes or persists under representation variation. The framework is compatible with sensitivity analysis (one could compute sensitivity indices over a binarized decision map), but it does not require or assume a continuous output metric.

7.3 Reproducibility in Machine Learning

Underspecification in machine learning pipelines [3] demonstrates that pipelines satisfying identical training criteria can yield predictors with divergent behavior under distribution shift. Henderson et al. [8] show that deep reinforcement learning results are sensitive to implementation details and hyperparameter choices in ways that standard reporting obscures. Bouthillier et al. [2] formalize variance accounting across machine learning benchmarks. Reproducibility checklists [12] encourage reporting of experimental details but do not enforce content-addressed provenance or deterministic replay.

Decision-valued mapping addresses a related but distinct problem. Rather than documenting variability in performance metrics across training runs or hyperparameter settings, it isolates representational variation as an independent variable and tracks its effect on discrete outcome identity under fixed snapshots and engines. The framework does not replace reproducibility checklists; it provides a lower-level infrastructure for testing whether specific outcomes are stable under specific representational changes.

7.4 Provenance and Workflow Systems

Provenance models such as W3C PROV [10] provide general-purpose vocabularies for recording the derivation history of computational artifacts. VisTrails [5] captures workflow provenance for scientific computations. MLflow [17] tracks experiments, parameters, and artifacts for machine learning pipelines. The Common Workflow Language [1] standardizes workflow definitions for reproducible execution. Resilient Distributed Datasets [16] track lineage for fault-tolerant distributed computation.

DecisionDB is narrower than these systems in scope and more specific in its invariants. It does not manage arbitrary workflows or track general-purpose provenance graphs. It enforces a specific structure: frozen snapshots, declared representation families, fixed engines, and equivalence policies that reduce raw output to discrete identities. The content-addressing scheme ensures that identical inputs always produce identical identifiers, and replay verification checks end-to-end consistency of the provenance chain. This specificity enables the decision-valued map as a queryable diagnostic object, which general-purpose provenance systems do not directly support.

7.5 Infrastructural Precedents

The approach follows a pattern observed in the development of foundational abstractions. Abstract data types [9] separated representation from observable behavior, enabling systems to evolve internally without collapsing external guarantees. Write-ahead logging [7] transformed durability from an ad-hoc property into an auditable, replayable state transition protocol. In both cases, the contribution was not a novel computational procedure but a diagnostic layer that made structural dependence explicit and testable. Decision-valued mapping extends this pattern to settings where discrete outcomes depend on representational choices in complex analytical pipelines.

8 Limitations

Discrete outcomes only. The framework applies to systems whose outputs can be reduced to discrete identities via a declared equivalence policy. Continuous outputs (e.g., probability distributions, regression surfaces) are out of scope unless such a reduction is explicitly defined. The choice of equivalence policy directly determines what counts as "the same outcome," and different policies applied to the same raw output will in general produce different decision maps.

Fixed snapshot and engine. All results assume a frozen snapshot and a fixed engine. Any change to the input data, model parameters, or execution logic constitutes a new analytical context. The framework does not track how decision maps evolve across snapshot or engine versions; each combination requires a separate analysis.

Empirical coverage. The reported sweeps cover two representation parameters (`neighbor_weight` at values 0.5 and 1.0; `second_order_weight` at values 0.25 and 0.5) applied to a single graph snapshot with a single origin-destination pair. The representation space is sampled at four points total. Unobserved regions of representation space remain unconstrained: persistence regions and boundaries identified here may not generalize to finer parameter grids, different origin-destination pairs, or different graph topologies.

Single domain. The empirical demonstration uses graph routing. The framework is designed to be domain-agnostic, but we have not validated it on classification, resource allocation, or other pipeline types. Applying the framework to a new domain requires defining an appropriate equivalence policy, which involves domain-specific judgment about what constitutes "the same outcome."

No causal claims. Observing that a boundary exists between two parameter values does not explain why it exists. The framework is diagnostic, not explanatory. It identifies where decision identity changes but does not attribute the change to any specific mechanism within the engine or the representation construction.

Scalability. The current implementation uses SQLite and has been tested with single-digit representation families. Scaling to large parameter grids (hundreds or thousands of representations) would require evaluation of storage, query performance, and sweep orchestration, which we have not performed.

9 Conclusion

We introduced decision-valued maps as a diagnostic object for characterizing how discrete outcomes depend on representational choices. The object is a mapping from representations to decision identities, evaluated under a fixed snapshot and engine, and materialized through content-addressed identifiers and immutable artifacts.

We described DecisionDB, a system that implements this framework through a five-stage sweep protocol, a five-table relational schema, and a replay verification procedure. In a graph routing demonstration, we observed that one representation parameter preserves route identity across its tested range while another induces a discrete route change. Replay verification confirmed that all persisted decision identifiers are deterministically recoverable.

The framework is limited by its restriction to discrete outcomes, its reliance on fixed snapshots and engines, and the narrow empirical coverage reported here. Extending the approach to finer parameter grids, additional domains, and cross-snapshot comparison are directions for future work.

The contribution is infrastructural. Making representational dependence observable does not improve outcomes, but it is a prerequisite for understanding when discrete outcomes can be trusted and when they cannot.

References

[1] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, John Kern, Dan Leehr, Hervé Ménager, et al. Common workflow language, v1.0. `https://www.commonwl.org/v1.0/`, 2016.

[2] Xavier Bouthillier, César Laurent, and Pascal Vincent. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

[3] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23:1–61, 2022.

[4] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[6] Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. Department of Statistics, Columbia University, 2013.

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

[8] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[9] Barbara Liskov and Stephen Zilles. Programming with abstract data types. In *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*, pages 50–59. ACM, 1974.

[10] Luc Moreau and Paolo Missier. PROV-DM: The PROV data model. W3C Recommendation, 2013. `https://www.w3.org/TR/prov-dm/`.

[11] Chirag J. Patel, Jay Bhatt, Atul J. Patel, and John P. A. Ioannidis. An environment-wide association study (EWAS) on type 2 diabetes mellitus. *PLoS ONE*, 10(7):e0132002, 2015.

[12] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021.

[13] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis: The Primer.* John Wiley & Sons, 2008.

[14] Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. Specification curve analysis. *Nature Human Behaviour*, 4(11):1208–1214, 2020.

[15] Sara Steegen, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. Increasing transparency through a multiverse analysis. *Perspectives on Psychological Science*, 11(5):702–712, 2016.

[16] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 15–28, 2012.

[17] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Engineering Bulletin*, 41:39–45, 2018.