On Decision-Valued Maps and Representational Dependence

Gil Raitses

Abstract

Complex analytical pipelines produce discrete outcomes whose dependence on representational choices is rarely made explicit. On a fixed graph snapshot with a fixed computational engine, sweeping one representation parameter preserves the discrete outcome while sweeping another produces a different outcome under identical query conditions. A decision-valued map records these relationships, treating the dependence of outcome on representation as an observable, storable object. This paper formalizes the concept and describes DecisionDB, a minimal infrastructure that logs, replays, audits such maps using content-addressed identifiers, immutable artifacts, declared equivalence policies. Deterministic replay recovers each decision identifier from its stored artifacts, establishing that the recorded context fully specifies the outcome. The contribution is infrastructural, establishing a system-level abstraction that partitions representation space into persistence regions and boundaries and treats decision reuse as a mechanically checkable condition.

1 Introduction

Analytical pipelines in scientific, engineering, policy settings routinely produce discrete outcomes such as 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, including 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, 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. The central object is 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, it becomes possible to observe directly which outcomes persist under representation changes and where boundaries form.

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

The approach is demonstrated on a graph routing problem. Fixing a graph snapshot and a shortest-path engine, two representation parameters that control edge-cost construction are swept. One parameter preserves decision identity across its tested range; the other induces a discrete

identity change at a specific threshold. Replay verification confirms that persisted identifiers are deterministically recoverable.

The contribution is an infrastructure for observing what happens to discrete outcomes when representations change.

2 Problem Scope

Decision-valued mapping addresses systems with the following structure. A *snapshot* s is a frozen, immutable slice of external inputs over a declared time window; any change to the world state produces a new snapshot. A *representation* $r \in \mathcal{R}(s)$ is a deterministic encoding of s, defined by explicit structural choices such as kernels, thresholds, weighting rules, aggregation policies; each representation is fully specified by a declared parameter set and generated by a versioned factory. An *engine* E is a fixed computational procedure that consumes a representation and produces raw output, with engine configuration and version held constant during analysis. An *equivalence policy* π is a declared rule that reduces raw engine output to a discrete *decision identity* $d \in \mathcal{D}$, defining when two raw outputs correspond to the same identity independent of incidental numerical differences.

The scope is diagnostic. The map characterizes when decision identity persists across representational variation and when it changes. It does 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, resource allocation under alternative aggregation rules.

3 The Decision-Valued Map

The central object of study is a mapping

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

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 are connected subsets of $\mathcal{R}$ over which f is constant; within a persistence region, representational variation preserves the outcome. Boundaries are loci in $\mathcal{R}$ where f changes value, separating two persistence regions with different decision identities. Fractures are boundaries where a small change in representation parameters induces a discrete identity change, indicating high sensitivity of the outcome to the representation.

The purpose of DecisionDB is to *materialize* f, evaluating it at declared points in $\mathcal{R}$, storing the results as immutable artifacts, then making the resulting map queryable, replayable, auditable. Figure 1 illustrates this pipeline.

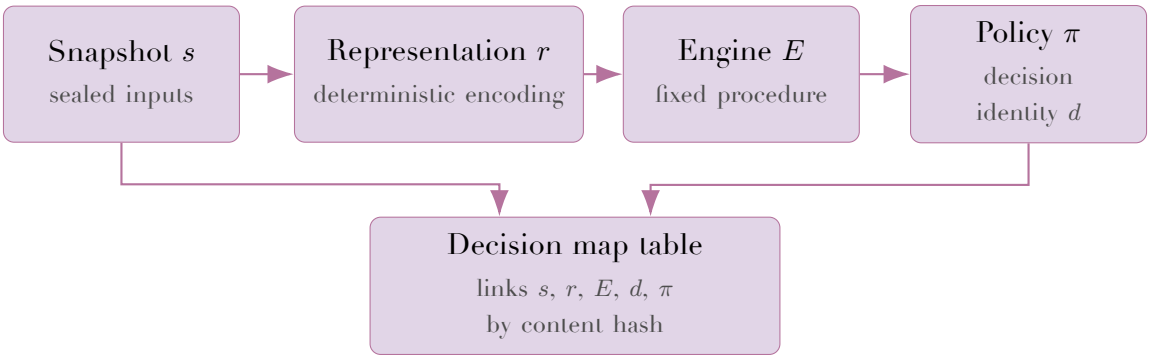

Figure 1: Decision-valued mapping pipeline. A snapshot is encoded into a representation, consumed by a fixed engine, then reduced to a discrete decision identity via an equivalence policy. The decision map table materializes these links using content-addressed identifiers. Each stage is immutable and content-addressed, so the full provenance chain from raw input to discrete outcome is recoverable at any later time.

4 Sweep Protocol

A *representational sweep* evaluates the decision-valued map f across a declared set of representations. The protocol proceeds through five stages, summarized in Table 1, that transform a frozen snapshot into a queryable decision map.

A sweep begins by freezing a snapshot and assigning it a content-addressed identifier. A representation family is then declared via a deterministic factory, and a sweep plan specifies the parameter grid, engine, equivalence policy. The engine executes independently for each representation, producing raw outputs stored as immutable artifacts. The equivalence policy reduces each output to a decision identity, and the decision map table records the links from snapshot and representation through engine run to decision identity. All artifacts are versioned and linked through content-addressed identifiers, so 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 as a Python dictionary, DecisionDB serializes it to canonical JSON with keys sorted alphabetically, no whitespace, arrays in declaration order, floats as strings, explicit version fields. It then computes the SHA-256 digest of the UTF-8 encoding, truncates to the first 16 hexadecimal characters, prepends a type-specific prefix. Identical content always produces identical identifiers, regardless of when or where the computation occurs.

5.2 Schema

The relational schema contains five core tables. Figure 2 shows their foreign-key relationships and Table 2 summarizes each table's role. Each table enforces content-addressed primary keys and foreign-key constraints that link the full provenance chain from snapshot through representation

Table 1: Five-stage representational sweep protocol.

	Stage	Inputs	Outputs
1	Freeze snapshot	World state, time window	Snapshot identifier
2	Declare representations	Snapshot ref, parameterization, factory version	Representation identifiers, one per parameter setting
3	Plan sweep	Parameter grid, engine name and version, equivalence policy	Sweep plan descriptor, itself content-addressed
4	Execute engine	Each representation independently	Run records, raw output artifacts stored as immutable files
5	Extract decisions	Raw outputs, equivalence policy π	Decision identifiers, decision map entries linking representations and runs to decisions

and engine execution to decision identity. All writes are append-only, scoped by experiment identifier, executed within transactions. Inserts use an insert-or-ignore strategy for idempotency, so 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, identifying which field of the raw output carries decision-relevant content such as the route node sequence. It also specifies a canonicalization rule that determines how to serialize the extracted content alongside a match rule that determines identity such as SHA-256 equality. The policy itself is content-addressed, so any change to the policy definition produces a new policy identifier and new decision identifiers downstream.

5.4 Replay Verification

Replay verification takes a persisted decision record, reloads the stored raw output and policy specification, recomputes the policy identifier, payload hash, decision identifier, then checks them against the persisted values. Because replay is read-only and writes no rows, a successful pass confirms that the content-addressing chain from raw output through policy application to decision identity is deterministic and self-consistent.

6 Empirical Demonstration

The approach is demonstrated on a graph routing problem. The goal is to show how the decision-valued map makes representational dependence observable in a concrete setting.

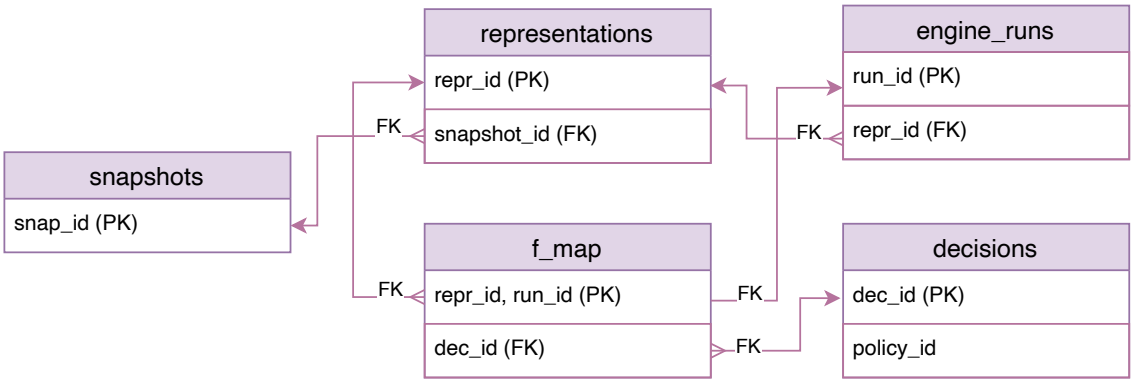

Figure 2: DecisionDB relational schema. Five tables form a content-addressed provenance chain. Foreign-key arrows indicate the direction of referential dependency, linking representations to their parent snapshot, engine runs to the representation consumed, the decision map table to representations, runs, decision identities.

Table 2: DecisionDB entity roles.

Table	Key prefix	Role
snapshots	snap	Immutable input state and its artifact manifest
representations	repr	Deterministic encodings of snapshots under a declared parameterization
engine runs	run	Execution records of the fixed engine on specific representations
decisions	dec	Discrete decision identities extracted by an equivalence policy
f map	composite	Materialized decision-valued map linking representations, runs, decisions

6.1 Setup

The demonstration fixes a directed graph with node set V where $|V| = 564$, edge set E, immutable edge attributes including baseline costs derived from geographic distance and a normalized stress metric. The snapshot is content-addressed and sealed before any engine execution. A single origin-destination query is fixed at start node 85 and end node 50.

Each representation encodes the graph's edge costs as a deterministic function of the fixed edge attributes. Two representation parameters control the cost surface. The first, neighbor weight, applies a weight to a neighbor-based cost component and is tested at 0.5 and 1.0. The second, second-order weight, applies a weight to a second-order cost component and is 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.

The engine is a fixed shortest-path solver using Dijkstra's algorithm [4], with configuration and version held constant across all runs. Execution times ranged from 0.5 to 1.4 milliseconds. The equivalence policy, version 1.0.0, performs exact match on the sorted node sequence of the computed

Table 3: 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

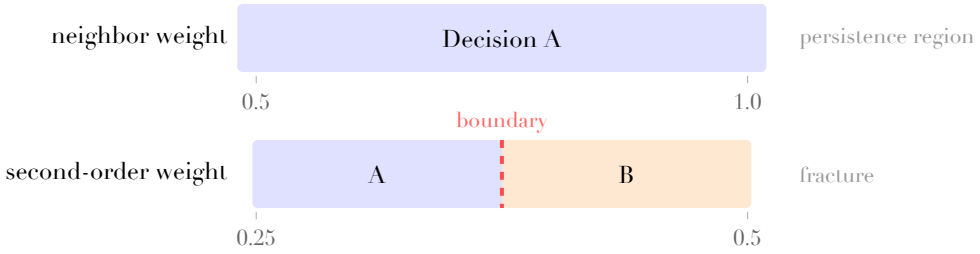

Figure 3: Identity persistence and fracture structure for the two representation parameters. The neighbor weight parameter spans a single persistence region in which both tested values produce Decision A, indicating decision identity is stable across this range. The second-order weight parameter spans two regions separated by a fracture, where decision identity changes from Decision A to Decision B between the values 0.25 and 0.5. The exact threshold is not resolved by this two-point sample.

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

In this instantiation, decision identities correspond to route equivalence classes, but the protocol applies to any discrete outcome produced by a fixed decision query.

The neighbor weight sweep varies the neighbor weight from 0.5 to 1.0 while holding second-order weight fixed at 0.25. Both representations yield the same route, Decision A, a 16-node path:

$$[85,\ 176,\ 463,\ 14,\ 404,\ 76,\ 406,\ 407,\ 223,\ 200,\ 311,\ 310,\ 314,\ 322,\ 323,\ 50].$$

Doubling the neighbor weight from 0.5 to 1.0 preserves decision identity. This parameter range constitutes a persistence region under the tested policy.

The second-order weight sweep varies the second-order weight from 0.25 to 0.5 while holding neighbor weight fixed at 0.5. At second-order weight 0.25, the engine produces Decision A. At second-order weight 0.5, the engine produces Decision B, a 14-node path:

$$[85,\ 411,\ 419,\ 422,\ 332,\ 204,\ 500,\ 369,\ 79,\ 402,\ 473,\ 502,\ 501,\ 50].$$

The route changes entirely, 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, where a small parameter change induces a qualitative change in the discrete outcome. Table 3 summarizes the sweep results; Figure 3 shows the persistence and fracture structure.

Table 4: Replay verification results.

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

6.3 Replay Verification

Replay verification is performed on a decision produced by the sweep. The procedure reloads the stored raw output and policy specification, recomputes all three identifying fields so as to check them against the persisted values. All recomputed values match exactly, as shown in Table 4 and Figure 4.

The replay writes no new rows and modifies no database state. Table counts before and after replay are identical, with 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.

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, provenance systems. Table 5 summarizes the key distinctions.

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 such as regression coefficients and p-values, then 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.

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

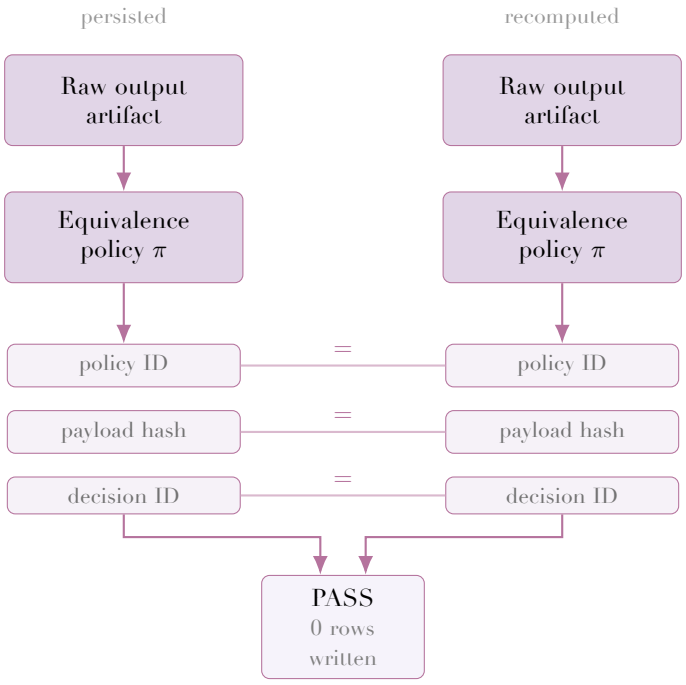

Figure 4: Replay verification pipeline. Both columns begin with the same stored raw output artifact. The persisted column records the identifiers that were computed during the original sweep. The recomputed column re-applies the equivalence policy to the stored artifact, independently regenerating the policy identifier, payload hash, decision identifier. All three pairs match exactly. The procedure writes no rows; a successful pass confirms that the content-addressing chain from raw output through policy application to decision identity is deterministically recoverable.

or compute sensitivity indices. Instead, it directly observes whether a discrete outcome changes or persists under representation variation. Decision-valued mapping is compatible with sensitivity analysis, since sensitivity indices could be computed 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. Decision-valued mapping complements reproducibility checklists by providing a lower-level infrastructure for testing whether specific outcomes are stable under specific representational changes.

Table 5: Summary comparison between decision-valued mapping and related approaches.

Dimension	Typical approaches	Decision-valued mapping
Object of analysis	Continuous effect estimates, variance, p-values	Discrete decision identities
Variation source	Model specs, preprocessing, hyperparameters	Representation parameters under fixed snapshot and engine
Output characterization	Distributions, sensitivity indices, specification curves	Persistence regions, boundaries, fractures
Infrastructure	Logging, provenance metadata, workflow graphs	Content-addressed, replayable decision maps

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, 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. Rather than managing arbitrary workflows or tracking general-purpose provenance graphs, it enforces a specific structure with immutable snapshots, declared representation families, fixed engines, equivalence policies that reduce raw output to discrete identities. The content-addressing scheme ensures that identical inputs always produce identical identifiers; 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

Decision-valued mapping applies to systems whose outputs can be reduced to discrete identities via a declared equivalence policy. Continuous outputs such as probability distributions and 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.

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. DecisionDB does not track how decision maps evolve across snapshot or engine versions; each combination requires a separate analysis.

The reported sweeps cover two representation parameters, neighbor weight at values 0.5 and 1.0 and 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, and unobserved regions remain unconstrained.

Persistence regions and boundaries identified here may not generalize to finer parameter grids, different origin-destination pairs, or different graph topologies.

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

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.

The current implementation uses SQLite and has been tested with single-digit representation families. Scaling to large parameter grids of hundreds or thousands of representations would require evaluation of storage, query performance, sweep orchestration, none of which has been performed.

9 Conclusion

Decision-valued maps provide a diagnostic way to make the dependence of discrete outcomes on representational choices explicit. The object is a mapping from representations to decision identities, evaluated under a fixed snapshot and engine, then materialized through content-addressed identifiers and immutable artifacts.

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

The approach is limited by its restriction to discrete outcomes, its reliance on fixed snapshots and engines, the narrow empirical coverage reported here. A natural extension is to densify the parameter sweep, apply the same protocol across additional domains, introduce a cross-snapshot comparison procedure that treats "same query, different snapshot" as an explicit axis of reuse admissibility.

The contribution of this work is infrastructural, introducing a diagnostic layer that records how discrete decisions arise from families of representations. Decision-valued maps expose stability and fracture directly, allowing downstream reuse to be conditioned on explicit representational context rather than on confidence or performance summaries alone. Decision reuse becomes a compatibility question that can be checked mechanically before deployment across tasks and timescales.

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.