UIM Theorem Frontier

Technical Companion and User Guide

What information survives a mathematical representation?

An executable exposition of representation fibers, exact obstruction, recovery, and registered proof exploration.

Kevin L. Brown
Independent Researcher

August 2026

Plugin release:	0.2.1
Architecture specification:	1.5
Graph Experiment specification:	UTF-1.0
Proof Experiment specification:	UPE-1.2
Prototype:	https://creationunified.com/uim-theorem_frontier

Abstract

UIM Theorem Frontier is an executable mathematical exposition for a precise question: when a mathematical object is replaced by a representation, what information survives, which properties can still be determined, and how can a reader inspect the proof boundary rather than merely read a fixed conclusion?

The central criterion is the feature-fiber equivalence recorded as UIM Theorem 6.5. For a feature map e and target predicate P, the property factors through the representation exactly when P is constant on every fiber of e. The result itself is elementary; the contribution of Theorem Frontier is the executable architecture that combines representation choice, exact fiber computation, adversarial obstruction search, bounded certification, exact recovery, registered proof dependencies, stress testing, and reproducible experiment identity in one scholarly instrument.

$$\boxed{P = p \circ e \quad \Longleftrightarrow \quad \forall X, Y : \; e(X) = e(Y) \Rightarrow P(X) = P(Y)}$$

A **fiber** is the set of source objects that receive the same representation value.

The public plugin now exposes three mathematical modes: **Feature Test**, **Recovery Test**, and **Proof Explorer**. Feature Test and Recovery Test use finite simple labeled graphs. Proof Explorer executes a frozen module for UIM Theorem 6.5 over a bounded finite mapped-set example. The system is deterministic, does not use AI-generated proof judgments, and does not promote incomplete or bounded computation into an unrestricted theorem.

The mathematical problem

Mathematics routinely replaces a structure by a reduced description: a graph by a degree sequence, a matrix by a spectrum, or a dynamical system by selected orbit statistics. Such representations can be useful precisely because they discard information. The logical question is whether the discarded information was required by the property or theorem under investigation.

Let D be a class of source objects and let

$$e : \mathrm{Ob}(D) \longrightarrow Z \tag{1}$$

be a representation. Let

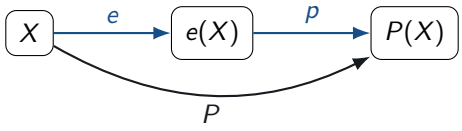
$$P : \mathrm{Ob}(D) \longrightarrow A \tag{2}$$

be the target property. We ask whether there exists a rule

$$p : Z \longrightarrow A \tag{3}$$

such that

$$P = p \circ e. \tag{4}$$



UIM Theorem 6.5 gives the exact criterion: such a p exists if and only if P is constant on every fiber of e. If two source objects collapse to the same representation while the target property distinguishes them, the representation cannot determine that property on any class containing both objects.

Novelty boundary. The fiber-constancy equivalence is elementary. Theorem Frontier does not claim to invent it. The contribution is the executable scholarly architecture: the reader can change mathematical information, recompute what collapses, obtain exact witnesses or certificates, inspect proof dependencies, and see the strongest conclusion justified by the resulting state.

Three mathematical modes

The v0.2.1 interface organizes the instrument into three distinct modes.

Mode	Core question	Current executable domain
Feature Test	Can the selected representation determine the selected target property?	Finite simple undirected labeled graphs, $1 \leq n \leq 6$.
Recovery Test	Can the original mathematical object be reconstructed from the representation?	Exact labeled adjacency encoding of the current finite graph object.
Proof Explorer	Why does the registered theorem follow, and what happens to its operational dependencies when the finite example changes?	Registered UIM Theorem 6.5 module over finite mapped sets of 2–8 source objects.

These modes are deliberately not interchangeable. Predicate-specific factorization is weaker than full recovery. A finite example can fail a theorem premise without falsifying the theorem. A registered theorem can justify a general conclusion only within its stated hypotheses and scope.

Feature Test: finite labeled graphs

For a fixed positive integer n, let

$$\mathcal{G}_n = \{\text{simple undirected graphs on the labeled vertex set } \{1, \dots, n\}\}. \tag{5}$$

Literal graph equality means equality of edge sets on this fixed labeled carrier. There are

$$|\mathcal{G}_n| = 2^{\binom{n}{2}} \tag{6}$$

such graphs. Thus $|\mathcal{G}_6| = 32{,}768$, small enough for exact browser enumeration in the current prototype.

The registered representation coordinates are:

ID	Feature	Definition / role
G-F01	$v(G)$	Number of vertices.
G-F02	$m(G)$	Number of edges.
G-F03	$\mathrm{degseq}(G)$	Sorted vertex-degree sequence.
G-F04	$t(G)$	Number of unordered vertex triples spanning a triangle.
G-F05	$c(G)$	Number of connected components.
G-F06	$a(G)$	Exact labeled adjacency bitstring; positive-control exact representation.

The target predicate registry includes connectedness, bipartiteness, triangle presence, Eulerian status under the frozen project convention, and tree status. Ordinary WordPress settings may control presentation and exposed demonstrations; they do not redefine the scientific registry.

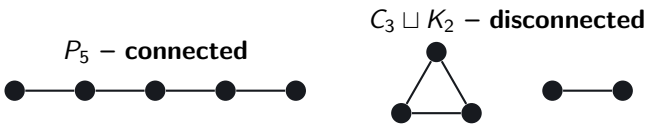
Canonical obstruction: degree sequence versus connectedness

Set

$$e(G) = \mathrm{degseq}(G), \qquad P_{\mathrm{conn}}(G) = \begin{cases} 1, & G \text{ connected}, \\ 0, & G \text{ disconnected}. \end{cases} \tag{7}$$

The adversarial search exhaustively rules out smaller graph sizes and finds its first exact obstruction at five vertices:

$$X = P_5, \qquad Y = C_3 \sqcup K_2. \tag{8}$$



Both graphs have degree sequence $(1, 1, 2, 2, 2)$, but their connectedness values differ. Therefore

$$e(P_5) = e(C_3 \sqcup K_2), \qquad P_{\text{conn}}(P_5) \neq P_{\text{conn}}(C_3 \sqcup K_2), \tag{9}$$

and no rule p using degree sequence alone can determine connectedness on any domain containing both witnesses.

Search boundary. The minimal-obstruction operation rules out every smaller vertex count, then terminates as soon as the decisive witness is found at $n = 5$. It does not need to exhaust the remainder of the five-vertex universe once non-factorization is already proved by the exact collision.

Complete fiber computation versus obstruction search

Compute representation fibers traverses the complete declared finite universe and builds the full fiber partition. If no heterogeneous fiber exists, no obstruction graph is displayed because there is no witness pair to draw.

Find smallest obstruction searches the declared hierarchy and may terminate early after a decisive heterogeneous fiber is found. Positive bounded certification requires complete traversal of the declared universe; interrupted or incomplete computation cannot be promoted to exhaustive factorization.

Theorem Frontier classifications and proof boundary

The system reports logical status rather than numerical confidence.

Status	Meaning	Permitted conclusion
TF-0	Invalid or incompatible experiment specification.	No mathematical conclusion.
TF-1	Exact heterogeneous fiber found.	Non-factorization on every class containing the witness.
TF-2	Complete finite enumeration; every fiber homogeneous.	Exact factorization on the declared bounded universe only.
TF-3	Separate registered theorem establishes factorization on the declared general class.	General conclusion only within that theorem's hypotheses and scope.

The key asymmetry is

$$\boxed{\text{one exact collision} \Longrightarrow \text{general factorization fails on every domain containing it}} \tag{10}$$

whereas

$$\boxed{\text{no collision in a completely enumerated } D_U \Longrightarrow \text{factorization proved on } D_U} \tag{11}$$

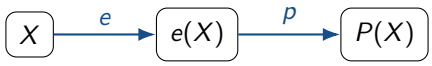
but, without an additional theorem,

$$\text{factorization on } D_U \not\Longrightarrow \text{factorization on an unrestricted larger class.} \tag{12}$$

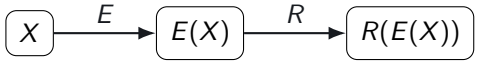
Quantifier boundary. More finite cases can strengthen a finite certificate, but they do not change a bounded quantifier into an unrestricted one. A general theorem requires an independently registered mathematical bridge.

Recovery Test: factorization is not reconstruction

Feature Test asks whether one property survives a representation. Recovery Test asks whether the original mathematical object can be reconstructed.



predicate-specific factorization



structural recovery: $R(E(X)) \simeq X$ (or $= X$ in the exact certificate)

A compressed feature may determine one predicate without retaining the entire graph. Conversely, exact labeled adjacency information permits direct reconstruction of the current finite labeled graph because the edge set is recoverable from the fixed bit ordering.

This distinction parallels the stronger UIM recovery calculus. UIM Theorem 6.1 shows that a natural recovery isomorphism forces faithfulness and reflects isomorphisms; Theorem 6.4 restricts transfer to an explicitly declared recoverable language; Theorem 6.7 gives a full, faithful, exactly recoverable realization for finite directed graphs. The current plugin uses a simpler finite simple undirected labeled graph domain and verifies its exact adjacency recovery directly.

Target leakage

A representation can directly contain the answer to a chosen predicate. For example,

$$P_{\text{conn}}(G) = 1 \quad \Longleftrightarrow \quad c(G) = 1 \tag{13}$$

under the declared nonempty graph convention. Such a factorization is mathematically valid but epistemically unsurprising. The interface therefore flags direct target information rather than presenting it as a discovered compression.

Proof Explorer: registered proofs become operational

Proof Explorer extends Theorem Frontier from representation testing into registered proof interaction. It is **not** an arbitrary theorem prover. The v0.2.1 release executes only frozen proof modules shipped with the reviewed plugin package.

The initial module is **UIM Theorem 6.5 – Feature-Fiber Criterion**. The theorem itself is general:

$$P = p \circ e \quad \Longleftrightarrow \quad P \text{ is constant on every fiber of } e. \tag{14}$$

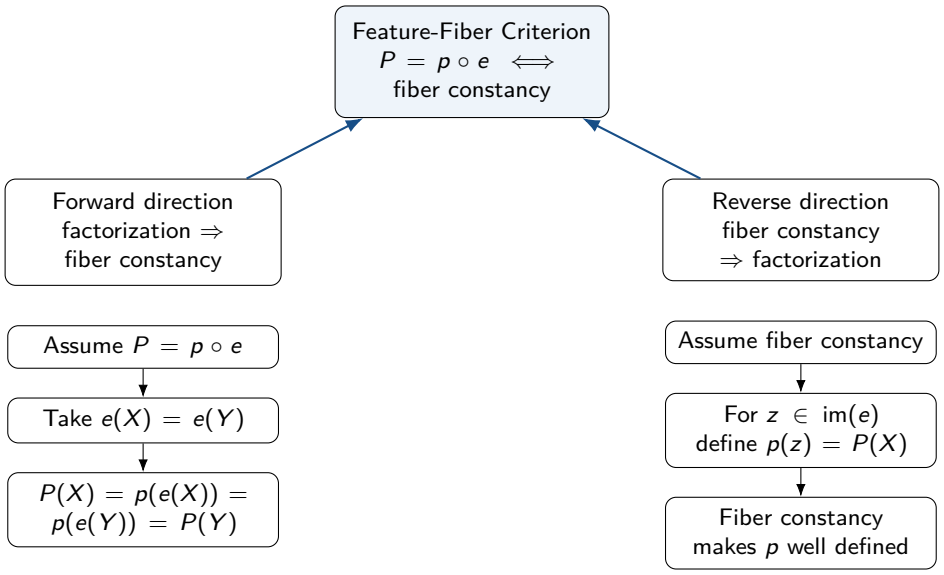
The executable browser module uses a bounded finite mapped-set example only so the fibers and obstruction/certificate can be computed exactly and immediately. Its current registered limit is 2–8 source objects.

The six Proof Explorer sections

Section	Interface heading	Function
P1	Choose the registered theorem	Displays the frozen theorem statement, plain-English meaning, scope, executable domain, hypotheses, and provenance.
P2	Change the finite example	Lets the reader alter $e(X)$, $P(X)$, and source-object count within registered bounds; live fibers recompute in a Worker.
P3	Follow the registered proof	Shows the hierarchical proof dependency map. Selecting a node exposes dependencies, consequences, and current-example status.
P4	Stress-test the argument	Runs registered operations that change the finite example, report what changed, and identify the first condition blocking the reverse construction of p.
P5	Inspect the justified result	Returns an exact finite certificate or exact obstruction while keeping the registered general theorem logically separate.
P6	Reproduce the proof experiment	Produces a deterministic Proof Experiment ID and fully versioned Proof Experiment link.

The proof has two frozen directions

The registered dependency structure follows the two implications of the theorem.



For values $z \notin \text{im}(e)$, $p(z)$ may be assigned arbitrarily because such values are never reached by e. The Proof Explorer registers this as a boundary step rather than silently omitting it.

Stress tests alter examples, not the theorem

The current module ships three registered presets: **Exact obstruction**, **Clean factorization**, and **Injective representation**. It also provides three registered stress operations:

- **Create exact obstruction**: deterministically creates a heterogeneous fiber.
- **Restore factorization**: makes each existing fiber constant by aligning property values within that fiber.
- **Make representation injective**: assigns every source object a different representation value, making every fiber a singleton.

These controls do not edit the theorem statement, proof nodes, or adapter code. They change only the bounded finite example. A heterogeneous fiber blocks the reverse construction because one representation value would require two different outputs from p.

> **Critical logical distinction.** If the current finite example violates fiber constancy, that example does not satisfy the premise needed for the reverse construction of p. This does *not* falsify Theorem 6.5. The theorem predicts exactly why factorization fails in that state.

Universal epistemic node states

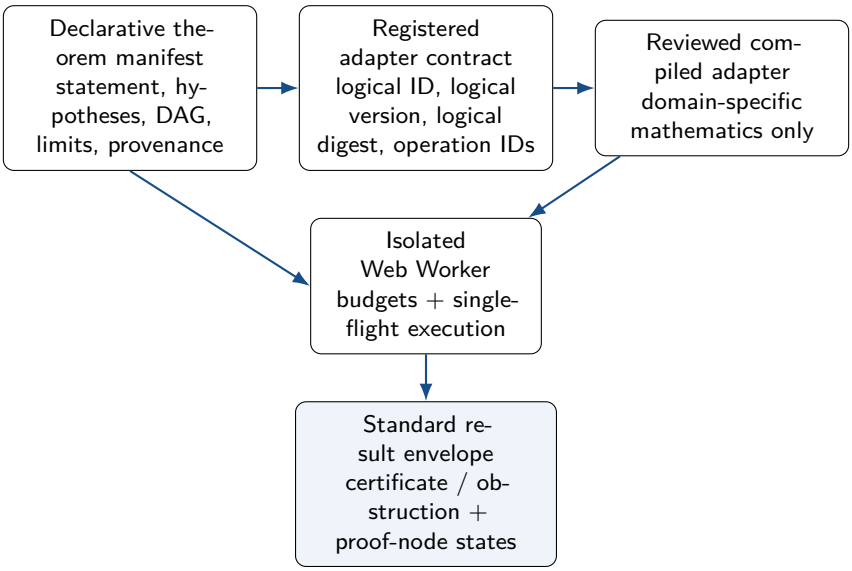
Domain-specific proof adapters may specialize the mathematics, but they do not redefine the meanings of proof status. Proof Explorer uses a fixed status vocabulary.

Status	Meaning
REGISTERED	Frozen theorem/proof content supplied by the reviewed module.
SATISFIED	The current executable state satisfies the node's condition.
FAILED	The current state provides an exact failure of the node's condition.
BLOCKED	A required upstream condition failed, so this downstream step cannot currently be instantiated.
NOT_EVALUATED	No evaluation has yet justified a state assignment.
OUTSIDE_DOMAIN	The requested state falls outside the registered theorem/module domain.

This result envelope lets the interface render proof dependencies without parsing or inventing domain-specific mathematics.

Registered proof-module architecture

Proof Explorer separates declarative scholarly structure from executable mathematics.



The registry cannot inject JavaScript. Operation names such as `fiber.proof.evaluate.v1`, `fiber.proof.certify.v1`, and `fiber.proof.stress.v1` resolve only to compiled functions already shipped in the reviewed release. There is no arbitrary theorem code input, remote mathematical import, `eval`, or AI-generated proof execution path.

The current finite-fiber module declares a maximum of 8 objects, 10,000 iterations, and a 2.5-second Worker budget. If the budget is exceeded, the required policy is:

> **COMPUTATIONAL LIMIT REACHED.** No mathematical conclusion is inferred from the incomplete computation.

A timeout is not relabeled as "no counterexample found."

Single-flight execution across the instrument

Proof Explorer parameter changes use a short debounce and hard-terminate obsolete Workers. Generation tokens reject stale responses. In v0.2.1, Feature Test, Recovery Test, and Proof Explorer also share execution ownership so competing mathematical Worker families do not run simultaneously in one instrument instance. Proof Explorer remains idle while hidden unless a Proof Experiment URL explicitly requires reconstruction.

Reproducibility: two experiment identities

Theorem Frontier uses separate canonical experiment identities for the graph engine and Proof Explorer.

Feature/graph Experiment ID

A Feature Test experiment records the finite scope, representation, predicate, graph-domain and kernel versions, and scientific registry versions. Canonical serialization produces a deterministic SHA-256 Experiment ID. A fully versioned Experiment link can be recomputed and checked; changing a mathematical control makes the previous experiment stale.

Proof Experiment ID

The Proof Experiment ID fingerprints the *mathematical logical state*, not incidental presentation or performance metadata. Under UPE-1.2 it binds:

- proof specification and epistemic-contract versions;
- theorem module ID, logical version, and independently recomputable logical digest;
- theorem ID, logical version, and logical digest;
- adapter ID, logical version, and logical-contract digest;
- registered operation IDs;
- exact finite example and relevant scientific state.

Implementation metadata – including plugin build, adapter implementation version/build, CSS, expanded proof nodes, and scroll position – is kept outside the mathematical hash and may be recorded separately in an execution receipt.

Logical version versus implementation version. A performance-only implementation improvement need not create a new mathematical experiment if the frozen logical contract is unchanged and conformance testing confirms identical behavior. A mathematical definition change requires a new logical version and therefore a different Proof Experiment identity.

Published logical versions are immutable. A correction to published mathematics is registered as a new logical version rather than silently modifying the old contract.

Deterministic core and validation boundary

The core mathematical conclusion is not produced by free-form AI text. The trusted computational layer includes graph enumeration and features, predicates, canonical serialization, fiber engines, frontier classification, recovery logic, theorem matching, registered proof manifests, adapter contracts, compiled proof operations, and Worker result validation.

The validation discipline distinguishes:

$$\boxed{\text{artifact exists} \neq \text{test performed} \neq \text{test passed.}} \tag{15}$$

The plugin therefore includes independent reference assets rather than allowing its browser code to certify itself by assertion.

For the v0.2.1 closure release, the package-level validation included PHP and JavaScript syntax checks, the unchanged 18-test independent graph reference suite, independent Proof Explorer registry/reference checks, compiled-adapter tests, Worker-contract tests, registry consistency, logical-digest recomputation, release-manifest SHA-256 verification, and ZIP integrity. Browser QA also exercised Feature Test, Recovery Test, Proof Explorer, keyboard tab navigation, mobile rendering, and Worker-driven interactions without uncaught plugin JavaScript errors.

Contribution and extension path

The verified production registry deliberately trades unrestricted code upload for security, provenance, and mathematical review. The governing rule is:

Open contribution, closed promotion.

A researcher may develop a theorem module against the published schema, but unreviewed executable mathematics does not enter the verified production registry directly.

Two contribution classes are distinguished:

Contribution	What changes	Promotion path
Manifest-only module	New theorem metadata, proof DAG, parameters, and provenance using an already registered mathematical adapter.	Schema/provenance review followed by inclusion in a reviewed registry/plugin release.
New mathematical adapter	Novel mathematical operations or domain-specific certification logic.	Mathematical specification, independent reference implementation, adversarial tests, security/determinism review, browser-budget review, and inclusion in a reviewed plugin release.

The intended lifecycle is

$$\text{DRAFT} \longrightarrow \text{SUBMITTED} \longrightarrow \text{UNDER REVIEW} \longrightarrow \text{REGISTERED} \longrightarrow \text{DEPRECATED}. \tag{16}$$

Deprecated logical versions remain addressable so historical Proof Experiment links retain meaning.

Potential later domains include deterministic transition systems. UIM Theorem 6.10 gives a natural mathematical basis for exact recovery and preservation of finite iterates, orbits, and reachability statements, while still respecting the boundary that exact representation does not automatically decide unbounded halting questions.

How to use the current prototype

A first session can follow three short demonstrations.

A. Feature Test: generate an exact obstruction

1. Open https://creationunified.com/uim-theorem_frontier and choose **Feature Test**.
2. In Section 1, use **Degree sequence** $\rightarrow$ **connectedness** with cumulative scope through $n = 5$.
3. In Section 2, run **Find smallest obstruction** $\leq n$.
4. In Section 3, inspect the generated P_5 and $C_3 \sqcup K_2$ witness, the shared degree sequence, and TF-1 strongest justified claim.
5. Switch to **Exact adjacency** $\rightarrow$ **connectedness** and run **Compute representation fibers**. No obstruction graphs should appear because no heterogeneous fiber was found; inspect the fiber counts and registered TF-3 conclusion instead.

B. Recovery Test: reconstruct the object

1. Choose **Recovery Test**.
2. Run the exact recovery certificate for a valid integer graph index.
3. Follow source object $\rightarrow$ exact adjacency representation $\rightarrow$ recovered object and confirm the recovery identity.

C. Proof Explorer: alter the finite proof state

1. Choose **Proof Explorer** and read P1 to distinguish the general theorem from the bounded executable module.
2. In P2, load **Exact obstruction**; observe a heterogeneous live fiber.
3. In P3, open the reverse-direction nodes and inspect why well-definedness of p is blocked by that fiber.
4. In P4, run **Restore factorization**; the finite state is changed so each fiber has one property value.
5. In P5, inspect the exact finite certificate and keep it logically separate from the registered general theorem.
6. In P6, generate the Proof Experiment ID and versioned Proof Experiment link only after the desired state has been recomputed.

What the instrument does not claim

UIM Theorem Frontier does not claim that UIM invented the elementary feature-fiber equivalence. It does not infer unrestricted theorems from finite enumeration, equate preservation of one predicate with recovery of an entire structure,

validate arbitrary user-supplied proofs, or solve open mathematical problems. Proof Explorer does not permit arbitrary proof code and does not use AI to manufacture mathematical conclusions.

The broader UIM monograph also distinguishes formal mathematical coherence from empirical or physical confirmation. Theorem Frontier is a mathematical communication and verification instrument; it does not by itself establish the empirical truth of the upstream physical ontology.

Why this is an executable exposition

For Feature Test the mathematical state is

$$S = (D_U, e, P). \tag{17}$$

Changing D_U, e, or P changes the mathematical experiment. The browser then recomputes

$$\boxed{D_U \longrightarrow e\text{-fibers} \longrightarrow P\text{-values} \longrightarrow \text{witness/certificate} \longrightarrow \text{strongest justified claim.}} \tag{18}$$

Proof Explorer adds a second operational layer:

$$\boxed{\begin{array}{c}\text{finite state} \longrightarrow \text{proof-node statuses} \longrightarrow \text{first blocked dependency} \\ \text{registered stress test} \longrightarrow \text{recomputed justification}\end{array}} \tag{19}$$

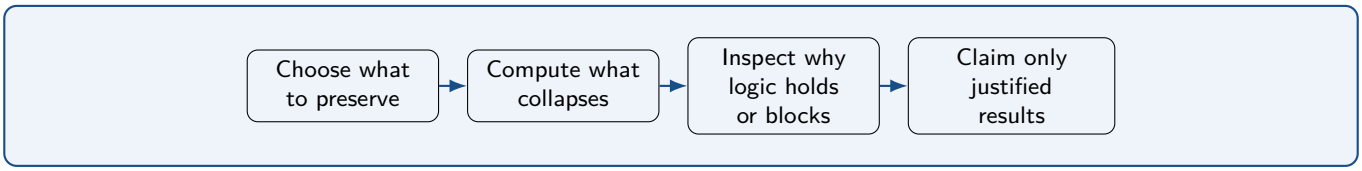
The interaction is therefore epistemic rather than decorative. It supports the scholarly loop

$$\boxed{\text{Understand} \longleftrightarrow \text{Experiment} \longleftrightarrow \text{Verify.}} \tag{20}$$

A conventional paper can state a theorem and explain its proof. Theorem Frontier lets a reader alter the mathematical information or finite proof state, recompute the consequences, inspect exactly where a conclusion becomes blocked or justified, and reproduce the resulting experiment.

Conclusion

Theorem Frontier begins from a modest but exact rule: if two objects receive the same representation while a target property distinguishes them, that representation cannot determine the property. The v0.2.1 architecture now connects that rule to three complementary forms of executable exposition.



The objective is not to make mathematics look interactive. It is to make mathematical information loss, recoverability, proof dependency, and the boundary of justified inference inspectable and reproducible.

Primary sources and implementation specifications

[1] Kevin L. Brown, *Unified Informational Mathematics: UIPO Foundations, Formal Semantics, Analytic Realization, and Recovery Theorems*, August 2026. Primary mathematical source for UIM Theorems 6.1, 6.4, 6.5, 6.7, and 6.10 and the representation-and-recovery calculus.

[2] Kevin L. Brown, *Unified Informational Mathematics and Ten Landmark Problems: An Adversarial Preservation-and-Recovery Stress Test with Formal Proof Development*, August 2026. Secondary source for proof/computation separation and bounded-versus-unrestricted proof discipline.

[3] Kevin L. Brown, *UIM Theorem Frontier - Mathematical and Functional Specification*, v1.5, August 2026. Closure architecture source for Feature Test, Recovery Test, registered Proof Explorer UPE-1.2, logical-digest reproducibility, shared execution ownership, structured provenance, and interface boundaries.

[4] *UIM Theorem Frontier Proof Module Contribution Pipeline*, August 2026. Implementation policy for manifest-only modules, reviewed mathematical adapters, immutable logical versions, and the open-contribution/closed-promotion model.