# Construction and Validation Architecture in delaunay

ADAM GETCHELL, The George Washington University, USA

**Author TODO.** Write the abstract in your own words.

Additional Key Words and Phrases: TODO author-supplied keywords

## 1 INTRODUCTION

- **Author TODO.** State the problem and motivation.
- **Author TODO.** Explain why proof-bearing construction and validation architecture matter for a scientific computational-geometry library.
- **Author TODO.** Summarize the central contribution in your own words.
- **Author TODO.** Preview the paper structure.

Applications such as Causal Dynamical Triangulation (CDT) require a robust and reliable triangulation library, capable of maintaining correct properties of the triangulation in simulations that may alter their structure millions of times per simulation. These invariant properties should be verifiable during construction and mutation, and violations of these properties should be detectable and diagnosable, so that fixable errors may be corrected and unfixable errors may be reported.

Delaunay is a Rust library for constructing and mutating triangulations with insertion, deletion, and bi-stellar k-flip operations, while maintaining a range of invariant properties important for the construction of Pseudo- and Piecewise-Linear (PL) manifolds, the evaluation of geometric predicates, and the embedding of triangulations in Euclidean, toroidal, and spherical spaces.

**Author TODO.** Qualify the preceding overview so the ordinary mutable Euclidean/toroidal workflows are distinguished from the separate bounded $S^2/S^3$ spherical prototype.

## 2 BACKGROUND AND DEFINITIONS

- **Author TODO.** Define the triangulation, simplex, ridge, facet, and realization terms used throughout the paper.
- **Author TODO.** Introduce PL-manifold terminology and the link condition.
- **Author TODO.** Introduce Euclidean, toroidal, and spherical realization models.
- **Author TODO.** Introduce Delaunay and future geometric-predicate terminology.

## 3 PROOF-BEARING CONSTRUCTION ARCHITECTURE

### 3.1 Vocabulary and API Roles

- **Author TODO.** Define a published owner as a domain value whose Rust type carries cumulative invariants. Suggested examples: `Tds`, `Triangulation`, and `DelaunayTriangulation`.
- **Author TODO.** Define a builder as a configurable public workflow that may coordinate several proof boundaries before returning an owner or a typed failure. Suggested examples: `TdsBuilder`, `TriangulationBuilder`, `DelaunayRefinementBuilder`, `DelaunayTriangulationBuilder`, and `DelaunayIncrementalBuilder`.

Author's address: Adam Getchell, adam@adamgetchell.org, The George Washington University, Washington, DC, USA.

- **Author TODO.** Define a draft as an unpublished representation attached to one promotion boundary. Suggested examples: `TdsDraft`, `TriangulationDraft`, and the private `DelaunayTriangulationDraft`.
- **Author TODO.** Define a workspace as private mutable algorithm state that may contain incomplete bootstrap, repair, retry, cache, or scheduling data. Suggested examples: `DelaunayBootstrapWorkspace` and `DelaunayBatchWorkspace`.
- **Author TODO.** Explain why the architecture avoids the ambiguous term "candidate": a proof-boundary value is a draft, while mutable algorithm state is a workspace.

### 3.2 Incremental Promotion Through Proof Layers

- **Author TODO.** Present the explicit-connectivity ladder: raw vertices and simplex specifications to `Tds` at Levels 1–2, then `Triangulation` at Levels 1–4, then `DelaunayTriangulation` at Levels 1–5.
- **Author TODO.** Explain that each promotion preserves established lower proofs and checks only the missing layer unless a cumulative diagnostic audit is explicitly requested.
- **Author TODO.** Describe why lower layers cannot infer connectivity using a higher-layer Delaunay policy. Suggested contrast: explicit simplex specifications at the TDS boundary versus point-set inference in a Delaunay builder.
- **Author TODO.** Describe the point-driven batch path through `DelaunayBatchWorkspace` and the final private `DelaunayTriangulationDraft`.
- **Author TODO.** Describe the incremental path through `DelaunayIncrementalBuilder`: bootstrap in a `DelaunayBootstrapWorkspace` containing `TdsDraft`, followed by ordered Levels 1–2, 3–4, and 5 promotion when the first maximal simplex exists.
- **Author TODO.** Explain why the public incremental type is a builder rather than a draft: it is long-lived, fluent, and coordinates multiple proof boundaries before and after the first maximal simplex.

### 3.3 Failure Atomicity, Recovery, and Fast Paths

- **Author TODO.** State the publication rule: success returns a valid owner for the promised layer; failure returns a typed error and never exposes an invalid published owner.
- **Author TODO.** Describe consuming promotions that return the original lower owner on failure so callers may retry with a different policy.
- **Author TODO.** Contrast strict, non-mutating promotion with explicitly selected canonicalization or repair modes that use rollback workspaces.
- **Author TODO.** Explain how construction-owned fast-path evidence is bound to the exact owner identity, structural generation, and topology so stale evidence cannot publish a stronger owner.
- **Author TODO.** Discuss the empty-complex boundary: empty drafts are trivially publishable when their layer invariants hold, while nonempty partial bootstrap state cannot be published.

## 4 CONSTRUCTION AND VALIDATION HIERARCHY

### 4.1 Level 1: Element Validity

- **Author TODO.** Describe what individual vertices, simplices, facets, and coordinates must prove locally.
- **Author TODO.** Explain what this level deliberately does not check.

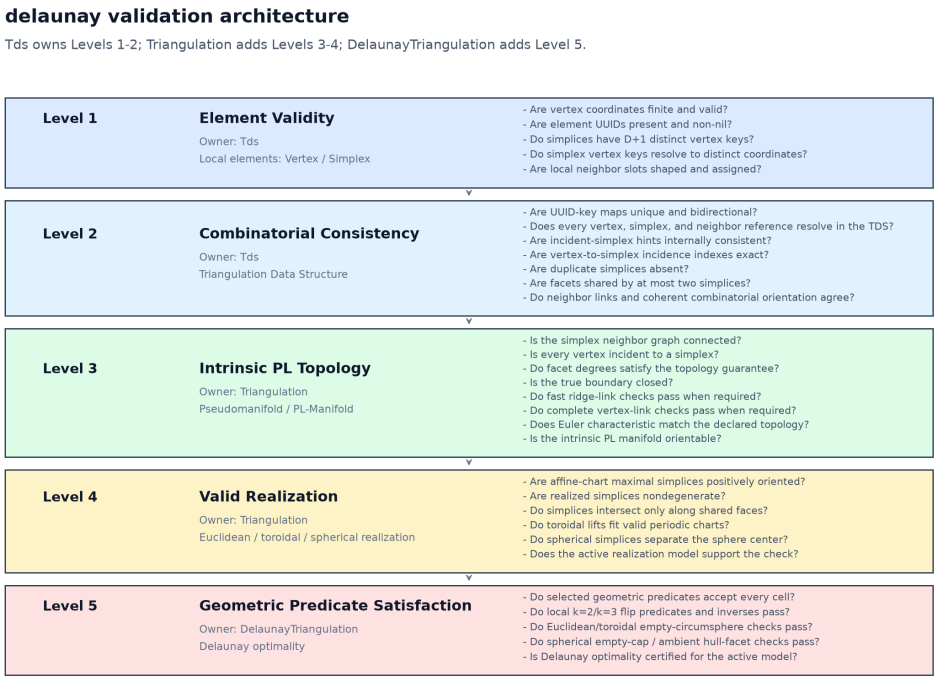


Fig. 1. Overview of the validation hierarchy in delaunay: `Tds` owns Levels 1–2, `Triangulation` adds Levels 3–4, and `DelaunayTriangulation` adds Level 5.

### 4.2 Level 2: Combinatorial Consistency

- **Author TODO.** Describe adjacency, incidence, neighbor symmetry, and TDS integrity.
- **Author TODO.** Distinguish coherent stored simplex orderings, including periodic quotient facets, from intrinsic orientability.
- **Author TODO.** Explain the separation from topology and geometry.

### 4.3 Level 3: Intrinsic PL Topology

- **Author TODO.** State the PL-manifold and boundary conditions.
- **Author TODO.** Discuss pseudomanifold and strict PL-manifold policy choices.
- **Author TODO.** Describe the 2D/3D intrinsic orientation witness and periodic quotient parity constraints.
- **Author TODO.** Explain why this layer is independent of Euclidean, toroidal, and spherical realizations.

### 4.4 Level 4: Valid Realization

- **Author TODO.** Use the terminology stack: Level 4 Valid Realization; Euclidean/toroidal valid affine-chart realization; spherical valid spherical realization; Level 5 Geometric Predicate Satisfaction / Delaunay Optimality.
- **Author TODO.** Describe valid realization in the active ambient model.
- **Author TODO.** Compare Euclidean/toroidal affine-chart realizations and spherical realizations.

- **Author TODO.** Keep positive geometric orientation distinct from Level 2 stored coherence and Level 3 intrinsic orientability.
- **Author TODO.** Explain why realization validity is a precondition for geometric predicates.
- **Author TODO.** Introduce two realized $d$-simplices with vertices $a_0, \ldots, a_d$ and $b_0, \ldots, b_d$, vertex labels $\ell_i$ and $m_j$, and shared-label set $S$, then explain the exact shared-face intersection program in Equation 1.

$$
\begin{aligned}
z^\star = \underset{\alpha_i, \beta_j \geq 0}{\text{maximize}} \quad & \sum_{\ell_i \notin S} \alpha_i + \sum_{m_j \notin S} \beta_j \\
\text{subject to} \quad & \sum_{i=0}^{d} \alpha_i a_i = \sum_{j=0}^{d} \beta_j b_j, \\
& \sum_{i=0}^{d} \alpha_i = 1, \qquad \sum_{j=0}^{d} \beta_j = 1.
\end{aligned}
\tag{1}
$$

- **Author TODO.** Explain the three exact outcomes: infeasibility proves the simplices disjoint; $z^\star = 0$ proves every feasible intersection point is confined to the shared face; and $z^\star > 0$ supplies barycentric weight on non-shared labels for a typed intersection witness.
- **Author TODO.** Explain how revised-simplex basis navigation replaces exhaustive active-set enumeration of the intersection polytope on the common path and removes the bottleneck exposed by the 4D and 5D Level 4 checks.
- **Author TODO.** Describe the filtered-exact hierarchy: bounding-box, separating-facet, orientation, and shared-face normal tests reject or certify easy pairs; floating-point simplex phases propose axes and bases; proved floating-point error bounds or exact rational primal/dual checks certify accepted results; and active-set enumeration remains the conservative fallback.
- **Author TODO.** Describe how exact finite-`f64` systems use the runtime-dispatched fraction-free Bareiss solver from `la-stack`, while general rational systems retain the `BigRational` fallback.
- **Author TODO.** Treat the broader independent randomized and degenerate 2D–5D agreement campaign and representative before/after benchmarks as v0.8.1 work tracked by issue #482.
- **Author TODO.** Do not present the bounded v0.8.0 4D and 5D slow-test probes as performance evidence. Coordinate any Level 5 all-violations performance claims with the separate v0.8.1 work in issue #483.

### 4.5 Level 5: Geometric Predicates

- **Author TODO.** Describe Delaunay as the first implemented predicate family.
- **Author TODO.** Explain how Euclidean, toroidal, and spherical Delaunay predicates differ.
- **Author TODO.** Leave room for regular, weighted, constrained, Gabriel, alpha, and future predicate families.

## 5 IMPLEMENTATION ARCHITECTURE

Table 1. Public validation API by owning layer.

| Level | Canonical public surface |
|---|---|
| 1 | `Vertex::is_valid` / `vertex_report`; `Simplex::is_valid` / `simplex_report` |
| 2 | `Tds::is_valid` / `structure_report`; `Tds::validate` certifies Levels 1–2 |
| 3 | `Triangulation::is_valid_topology` / `topology_report` / `orientation_witness`; `Triangulation::validate` certifies Levels 1–3 |
| 4 | `Triangulation::is_valid_realization` / `realization_report`; `validate_realization` certifies Levels 1–4 |
| 5 | `DelaunayTriangulation::is_valid_delaunay` / `delaunay_report`; `DelaunayTriangulation::validate` certifies Levels 1–5 |

- **Author TODO.** Relate each public builder terminal and private draft to the proof-bearing owner it publishes.
- **Author TODO.** Distinguish builder options, draft promotion state, workspace mutation state, and owner-level validation methods in the API map.
- **Author TODO.** Map the five validation levels onto the Rust API surface.
- **Author TODO.** Describe the role of fast-fail checks, diagnostics, reports, and cumulative validation.
- **Author TODO.** Explain construction-time validation policy choices such as `ExplicitOnly` and `OnSuspicion`.
- **Author TODO.** Describe how spherical topology fits as a coordinate/metric backend rather than a new simplex dimension.

## 6 DIAGNOSTICS AND REPRODUCIBLE FIGURES


Fig. 2. Level 1 Element Validity (Tds, cumulative Levels 1–2). **Author TODO.** Describe the validation witnesses shown in this panel.

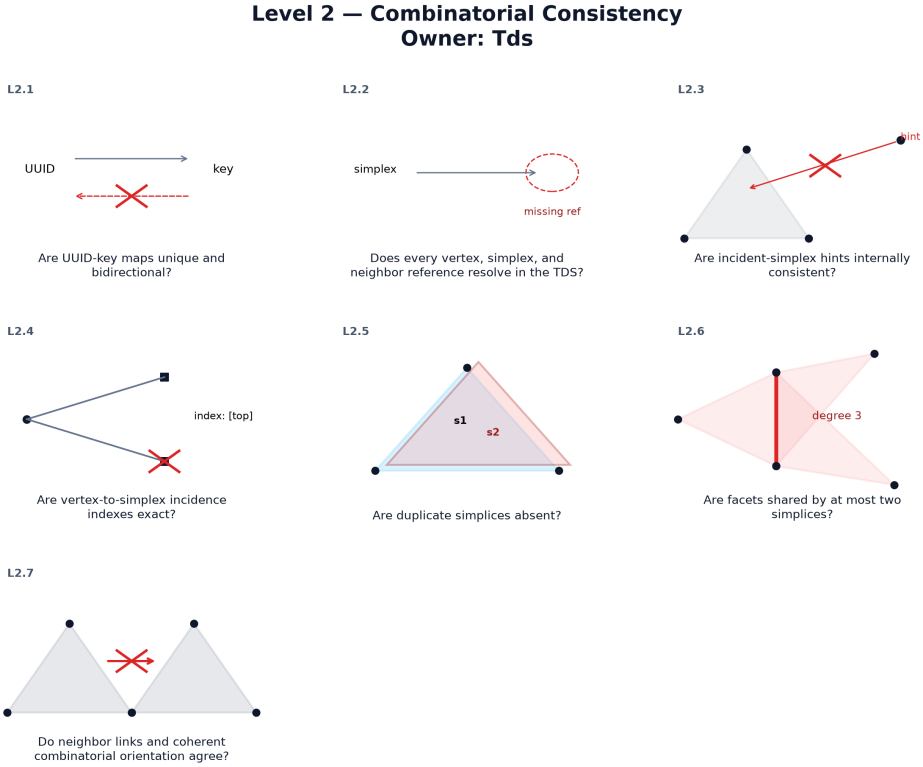


Fig. 3. Level 2 Combinatorial Consistency (Tds, cumulative Levels 1–2). **Author TODO.** Describe the validation witnesses shown in this panel.

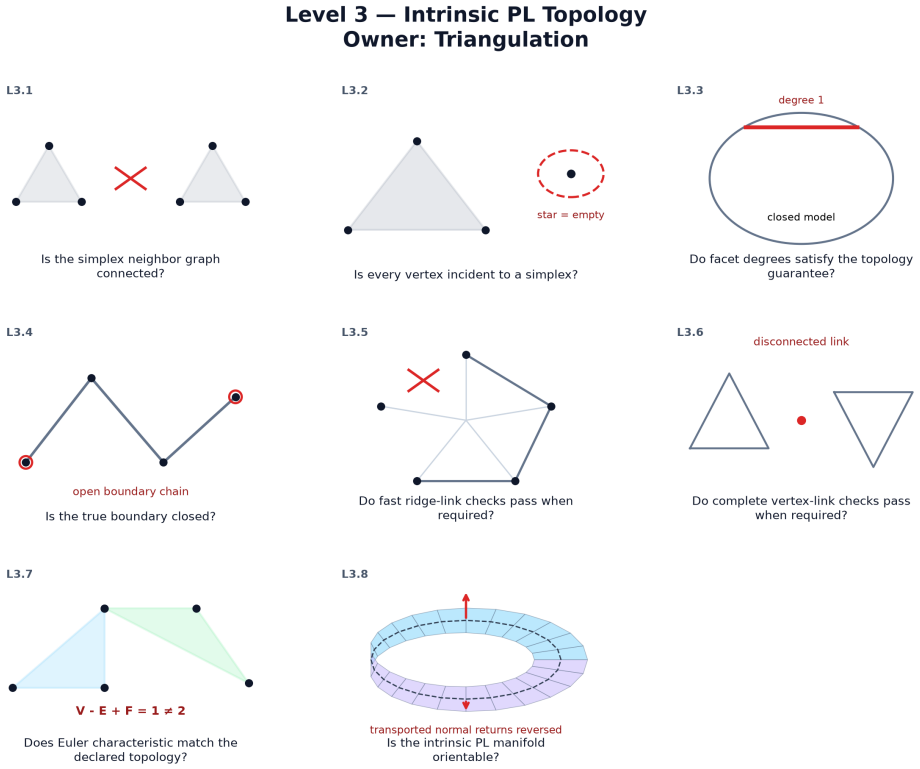


Fig. 4. Level 3 Intrinsic PL Topology (`Triangulation`, cumulative Levels 1–4). **Author TODO.** Describe the validation witnesses shown in this panel.

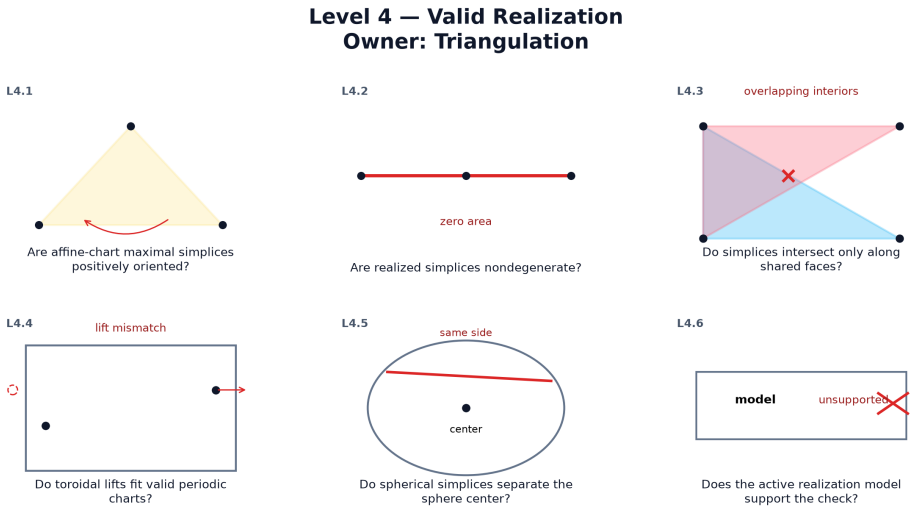


Fig. 5. Level 4 Valid Realization (`Triangulation`, cumulative Levels 1–4). **Author TODO.** Describe the validation witnesses shown in this panel.

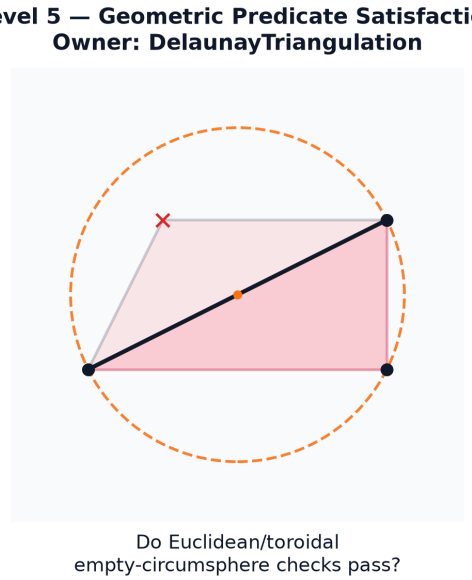


Fig. 6. Level 5 Geometric Predicate Satisfaction (`DelaunayTriangulation`, cumulative Levels 1–5). **Author TODO.** Describe the validation witness shown in this panel.

- **Author TODO.** Explain how the validation notebook generates paper figures.
- **Author TODO.** Describe representative diagnostics without duplicating API documentation.
- **Author TODO.** Connect figure generation to the reproducibility story.

## 7 RELATED WORK

- **Author TODO.** Place PL topology references and Pachner/Lickorish results.
- **Author TODO.** Place Euclidean robust-predicate and Delaunay references.
- **Author TODO.** Place spherical Delaunay and Riemannian-manifold references.
- **Author TODO.** Clarify which claims are architectural observations rather than literature claims.

## 8 DISCUSSION AND FUTURE WORK

- **Author TODO.** Discuss extensibility to additional realization models and predicate families.
- **Author TODO.** Discuss whether the Draft/Builder/Workspace vocabulary generalizes to regular, weighted, constrained, and non-Delaunay triangulation workflows.
- **Author TODO.** Discuss limitations of the current implementation.
- **Author TODO.** Identify open validation and benchmarking questions.

## 9 CONCLUSION

- **Author TODO.** Write the conclusion in your own words.

## ACKNOWLEDGMENTS

The author thanks Steven Carlip for discussions and guidance over the years on Causal Dynamical Triangulation, which motivated this work, and for encouragement and questions on this library, which motivated this paper.

## REFERENCES

[1] Manuel Caroli, Pedro M. M. de Castro, Sébastien Loriot, Olivier Rouiller, Monique Teillaud, and Camille Wormser. 2010. Robust and Efficient Delaunay Triangulations of Points on or Close to a Sphere. In *Experimental Algorithms (Lecture Notes in Computer Science, Vol. 6049)*. Springer, Berlin, Heidelberg, 462–473. https://doi.org/10.1007/978-3-642-13193-6_39

[2] Herbert Edelsbrunner. 2001. *Geometry and Topology for Mesh Generation*. Cambridge University Press, Cambridge, United Kingdom. https://doi.org/10.1017/CBO9780511530067

[3] Greg Leibon and David Letscher. 2000. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *Proceedings of the sixteenth annual symposium on Computational geometry*. ACM, Clear Water Bay Kowloon Hong Kong, 341–349. https://doi.org/10.1145/336154.336221

[4] W. B. R. Lickorish. 1999. Simplicial moves on complexes and manifolds. https://doi.org/10.48550/arXiv.math/9911256 arXiv:math/9911256.

[5] Udo Pachner. 1991. P.L. Homeomorphic Manifolds Are Equivalent by Elementary Shellings. *European Journal of Combinatorics* 12, 2 (1991), 129–145. https://doi.org/10.1016/S0195-6698(13)80080-7

[6] Jonathan Richard Shewchuk. 1997. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry* 18, 3 (1997), 305–363. https://doi.org/10.1007/PL00009321