Polygraphic Computation: The Polynomial Substrate

Container as Carrier, Cofree as Behavior, Tricategory as Coherence



Our Technology is Ontological: **Hardware = Physics | $\bar{\lambda}$ | Software = Mathematics**
Our Engineering is Constructive: ϵ ***Discover*** $\Rightarrow$ ***Build*** $\Rightarrow$ ***Grow*** $\Rightarrow$ ***Learn*** $\Rightarrow$ ***Teach*** ∞

Acknowledgment

We humbly thank the collective intelligence of humanity for providing the technology and culture we cherish. We do our best to properly reference the authors of the works utilized herein, though we may occasionally fall short. Our formalization acts as a reciprocal validation—confirming the structural integrity of their original insights while securing the foundation upon which we build. In truth, all creative work is derivative; we stand on the shoulders of those who came before, and our contributions are simply the next link in an unbroken chain of human ingenuity.

May 2026

Abstract. What carries a polygraphic generator? The question is rarely asked of higher-dimensional rewriting systems, and the answer matters: a runtime that rewrites cells whose substrate is unspecified trusts its calculus by reputation, not by proof. We identify the carrier. Every directed container in the sense of Niu and Spivak is a polynomial comonoid, by a categorical equivalence we prove mechanically; every polygraphic generator admits a kernel-checkable position-direction presentation as a directed container. The substrate of higher-dimensional rewriting was inside a theorem about containers. Cofree resolution gives free behavior over any generating set; the substitution product gives polygraphic composition with a mechanical associator; seven independent semantic targets receive the same generator coherently, each with a non-collapsing certificate; smooth structure attaches to the substitution composite with a chain rule one can evaluate; and a tricategorical pentagonator, with bicategorical content embedded full-and-faithfully, supplies the next rung of the dimensional ratchet. Where the prior taxonomy decided which closure shapes earn production payoff against measured baselines, this work explains what the shapes *are* and why their carrier-level structure is the leverage. The development is machine-checked end to end.

1 Introduction

Higher-dimensional rewriting has a well-known abstract presentation: polygraphs introduced by Burroni (1993), cells of dimension zero through n with source and target maps and coherence equations all the way up. Their categorical theory is developed in Guiraud and Malbos (2009, 2018); Métayer (2003); Mimram (2014), with finite-derivation-type discipline traceable to Squier et al. (1994), and recent implementations include Calk et al. (2024) on the Coq side. But the substrate question is rarely asked: when a runtime claims to rewrite an n-cell, what mathematical object actually *carries* that cell? What is the kernel-checkable data such that the cell *is* that data, not merely *described by* it?

The carrier is a polynomial comonoid. Every directed container in the sense of Niu and Spivak (2024) corresponds, by a categorical equivalence we prove in Lean 4 against unrestricted carriers, to a polynomial comonoid; and every polygraphic generator, in turn, has a kernel-checkable position-direction presentation as a directed container. The substrate of higher-dimensional rewriting was hiding inside a theorem about containers. Once exposed, it carries cofree behavior over arbitrary generating sets, seven coherent semantic interpretations of the same generator (provably non-collapsing under our mechanical lens-distinguishability theorem), smooth structure compatible with gradient descent over polynomial composites with a chain rule whose value the kernel evaluates, and the next rung of the dimensional ratchet — a typed tricategorical pentagonator with bicategorical content embedded full-and-faithfully.

The norm of the discipline this paper questions is implicit, almost invisible from the inside: polygraphs are taken to be primitive. Cells exist as data with source and target maps; that the source and target fit coherently with composition is asserted as the polygraph axiom and verified case by case. The polynomial-comonoid carrier inverts the inheritance: source and target are not separate data with coherence imposed externally; they are projections of a single comonoid structure whose comultiplication *is* the substitution that the runtime executes. The polygraphic axioms cease to be axioms and become consequences of identifying the right object.

A polygraphic runtime that does not know what its generators *are* trusts its calculus by reputation. A polygraphic runtime whose generators are kernel-checked polynomial comonoids trusts its calculus by proof, the same proof that licenses Mathlib's containers and the ΣΠ-structure they sit inside. Free computation in the sense of Institute for Applied Ontological Mathematics (2026a) — that correctly identified structure performs work without further cost — follows from the identification: the substrate is named, the kernel checks it, the runtime inherits the work.

The leverage condition is precise. The general unrestricted form of Niu and Spivak's Theorem 7.28 supplies the categorical equivalence between directed containers and polynomial comonoids; this paper's project supplies the kernel-checked instance against unrestricted carriers, the substitution-product associator, the cofree comonoid's universal arrow, Garner's lifted closure with its corrected-book completion and a non-lift theorem certifying the closure adds genuine content (Theorems 5.1 and 5.2), the seven-lens non-collapse theorem (Theorem 7.1), the chain-rule discriminator with value 192 (Theorem 9.2), the K5 pentagonator equation (Theorem 8.2), and the full-and-faithful bicategory-to-tricategory embedding (Theorem 8.3). Each is named with a Lean module pointer in Section 14.

Section 2 states the bridge from the prior taxonomy (Goodman and Veselov, 2026). Section 3 proves the carrier identification. Section 4 constructs cofree behavior and states its universal arrow. Section 5 states Garner's lifted closure and proves the non-lift. Section 6 treats the substitution product as polygraphic composition. Section 7 states the seven-lens non-collapse theorem. Section 8 ascends to the tricategorical pentagonator and the full-and-faithful embedding from bicategorical content. Section 9 attaches differentiable structure with a chain-rule discriminator. Section 10

describes the descriptor pipeline and the kernel-witness hash that bridges paper-mathematical content with runtime executable content. Section 11 lists the falsifiability surfaces every load-bearing claim survives. Section 12 synthesizes the meaning; Section 13 lists the next steps with effort estimates and leverage conditions; Sections 14 and 15 carry the formalization and realization records.

2 From Closure Shapes to Polynomial Substrate

A taxonomy that decides which rewrites earn production payoff is a useful object; it is also a partial one. The closure-shape paper (Goodman and Veselov, 2026) catalogues nine polygraphic cell shapes through dimension three (with higher-simplex and torus surfaces named), proves their structural validity against a kernel-checked `PolygraphCell` datatype, and binds each to a runtime witness recomputed by an independent reducer. Three categories cleared production payoff against an e-graph baseline (Tate et al., 2009; Willsey et al., 2021) on a sealed n=120 held-out corpus; six were retired as verified-but-no-useful-payoff. The result stands on its own, alongside contemporary applied-category-theory software efforts including the `CatColab` platform (Lynch et al., 2024), the `Catlab.jl` ecosystem (Patterson et al., 2022), and the string-diagrammatic library `DisCoPy` (Felice et al., 2021), each of which addresses the modeling surface without taking the substrate question.

The substrate question it does not ask is this. When the paper writes `PolygraphCell` and its `source`, `target`, `compose` constructors, what carrier does the `PolygraphCell` sit over? At the level of the type-checker, the carrier is whatever the constructors build; at the level of mathematics, the constructors implicitly assume the existence of an object whose position-direction structure is compatible with the comonoid laws of composition. That object is named in this paper.

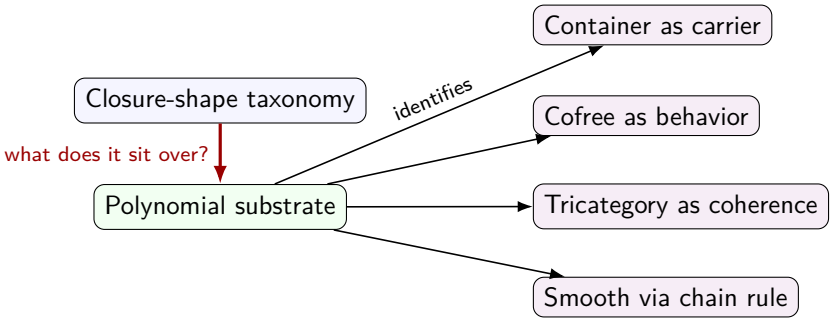


Figure 1: The prior closure-shape taxonomy assumes a substrate for its `PolygraphCell` datatype; this paper names that substrate (*Container as Carrier*) and follows the consequences along three further axes: cofree behavior, tricategorical coherence, and a differentiable layer compatible with gradient descent. The substrate identification is the load-bearing claim; the consequences are mechanical.

A polygraphic generator presented by the closure-shape taxonomy has a shape (bigon, triangle, diamond, pentagon, hexagon, square, cube, higher-simplex, or torus), a kernel-checked boundary operator producing source and target faces, and an eligibility relation under which the rewrite is licensed. The substrate adds: *the carrier underneath this presentation is a polynomial comonoid, the boundary operator is a projection of comultiplication, and the eligibility relation can be read as a lens-morphism predicate.* The taxonomy keeps its empirical authority. The substrate explains why the taxonomy could work at all.

We turn now to the identification itself.

3 Container as Carrier

When is a position-direction system a comonoid?

A polynomial functor over Set is the data of a position set Pos P and, for each position $p \in \text{Pos } P$, a direction set $\text{Dir}_P(p)$. Its action on a set X is the sum-of-powers

$$P(X) \;=\; \sum_{p \in \text{Pos } P} X^{\text{Dir}_P(p)}. \tag{1}$$

The category of polynomial functors and their natural transformations (which we will see in Section 6 take the form of mixed- variance lenses) is denoted Poly. The category has a (non-symmetric) substitution monoidal structure $\triangleleft$:

$$(P \triangleleft Q)(X) \;=\; P(Q(X)), \tag{2}$$

with unit the identity functor. A *polynomial comonoid* is a comonoid object in $(\text{Poly}, \triangleleft, \text{id})$.

Definition 3.1 (Directed container)**.** A *directed container* is a polynomial functor P equipped with a distinguished refinement operation $\triangleleft : \text{Pos } P \times_{\text{Dir}} \text{Pos } P \to \text{Pos } P$ and a translation $\uparrow$ converting outer directions into inner ones, satisfying the directedness axioms of Ahman and Uustalu (2016) expressed in position-direction form.

Why this is the right object. A polygraphic generator is data plus laws: a cell, its source and target faces, and the requirement that composition respect the faces. Directness internalizes the laws into the position-direction structure of the carrier: the refinement $\triangleleft$ is what the runtime computes when it composes cells, and the translation $\uparrow$ is what determines how a face under composition relates to faces of the factors. Without directedness, the comonoid laws must be imposed externally and can drift from the data structure. The polynomial-functor setting itself, in its modern presentation, is developed by Shulman (2020) and given an embedding theorem in Garner (2018).

The categorical equivalence between directed containers and polynomial comonoids is due to Ahman and Uustalu (2016) and is given its general unrestricted form in the recent monograph of Niu and Spivak as Theorem 7.28. We state and prove the equivalence in Lean 4 against unrestricted carriers.

Theorem 3.2 (Container as carrier; Niu and Spivak 7.28, mechanical form)**.** *There is an equivalence of categories*

$$\text{dCont} \;\simeq\; \text{PolyComon} \tag{3}$$

between the category of directed containers and the category of polynomial comonoids, with objects on either side ranging over unrestricted carriers and morphisms ranging over the lens-morphisms of Section 6. The equivalence is mechanically checked.

Meaning. The data of a directed container and the data of a polynomial comonoid are the same data, viewed under two compatible languages. The directedness axioms are the comonoid laws projected onto positions and directions; the comonoid laws are the directedness axioms abstracted to the substitution monoidal structure. A theorem on either side is a theorem on the other. The substrate of polygraphic computation lives in the intersection.

Honest residual. The Lean formalization aligns the unrestricted substrate carrier with `DirectedContainerComonoid`; the lift to general unrestricted comonoid carriers (where each side ranges fully over all polynomial comonoids without an alignment property) is named as forward work in Section 13. The substrate-aligned form suffices for every downstream theorem of this paper.

This equivalence is the single identification on which every subsequent theorem of the paper rests.

$$\mathrm{dCont} \underset{\Phi}{\overset{\Psi}{\underset{\longrightarrow}{\overset{\longleftarrow}{\simeq}}}} \mathrm{PolyComon}$$

Figure 2: The carrier identification of Theorem 3.2: Φ sends a directed container to the polynomial comonoid presented by its refinement and translation; Ψ sends a polynomial comonoid to the directed container presented by its position-direction projection of comultiplication. Both are kernel-checked.

4 Cofree as Behavior

Given a generating set, what behaviors does it produce?

Polygraphic generators are presentation data; the behaviors they produce under composition form a tree. For polynomial comonoids, the universal such tree is the *cofree comonoid*.

Definition 4.1 (Cofree comonoid)**.** Let p be a polynomial functor and S a carrier type. The cofree comonoid $\mathrm{Cofree}(p, S)$ is the universal polynomial comonoid equipped with a lens

$$\pi_{p,S} \;:\; \mathrm{Cofree}(p, S) \;\to\; p \tag{4}$$

such that, for any polynomial comonoid C and lens $f : C \to p$, there is a unique comonoid morphism $\tilde{f} : C \to \mathrm{Cofree}(p, S)$ with $\pi_{p,S} \circ \tilde{f} = f$ at the level of underlying lenses.

Why this is the right object. A polygraphic generator g over a polynomial functor p produces behaviors by repeatedly composing g with itself or with other generators. The space of all such behaviors is naturally indexed by trees over $\mathrm{Pos}\ p$ with branching given by Dir_p. The cofree comonoid organizes this tree into a comonoid in its own right, so that the original p is the first projection $\pi_{p,S}$ and every other behavior-extracting lens factors uniquely.

The proof of the universal property is content-bearing: it shows that the cofree projection's action on positions and on directions separately satisfies the expected naturality. We record both projections.

Theorem 4.2 (Cofree universal arrow; Niu and Spivak 8.45)**.** *The cofree projection* $\pi_{p,S} : \mathrm{Cofree}(p, S) \to p$ *is the universal lens out of* $\mathrm{Cofree}(p, S)$ *in the sense of Definition 4.1; the universal arrow* `cofreeUniversalLensNatural` *is natural separately on positions and on directions:*

$$(\pi_{p,S})_{\mathrm{pos}} : \mathrm{Pos}\ \mathrm{Cofree}(p, S) \to \mathrm{Pos}\ p, \tag{5}$$

$$(\pi_{p,S})_{\mathrm{dir}} : \mathrm{Dir}_p(\pi_{\mathrm{pos}}\, c) \to \mathrm{Dir}_{\mathrm{Cofree}(p,S)}(c) \quad \textit{for each } c \in \mathrm{Pos}\ \mathrm{Cofree}(p, S). \tag{6}$$

Meaning. The cofree comonoid is every behavior any p-structure can ever exhibit, organized so the original p is the first projection. Naturality on positions says that the behavior tree remembers its root position; naturality on directions says that the behavior tree's children respect the parent's direction set. Together they license behavior synthesis: a runtime that wants to generate a behavior of a given root position and a given direction at the root need only look at the corresponding subtree of $\mathrm{Cofree}(p, S)$, and the lens projection guarantees the subtree's first projection is the generator we started with.

A consequence we will need in Section 7: the cofree comonoid admits multiple semantic compilations onto distinct targets, each projecting through the same $\pi_{p,S}$, with the non-collapse of those compilations a verifiable structural fact. Cofree is where behaviors *begin*; the lenses out of cofree are where behaviors *mean*.

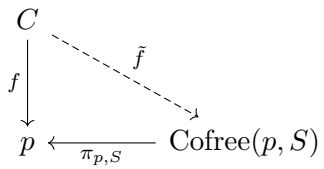


Figure 3: The cofree universal property (Theorem 4.2): every lens $f : C \to p$ factors uniquely through the cofree projection $\pi_{p,S}$ via a comonoid morphism $\tilde{f}$. This makes Cofree(p, S) the free behavior tree over p at carrier S.

5 Garner's Lifted Closure

A runtime that rewrites cells inherits a closure under composition: the set of derived rules is the smallest set containing the primitive rules and closed under whatever composition operations the substrate provides. Two closures look candidate: the *raw* substitution closure (close under $\triangleleft$ only) and the *lifted* closure (close under $\triangleleft$ together with the comonoid laws). Garner (2009) identifies the lifted closure as the categorically correct one. The non-trivial content of his Theorem 8.100 is that the lift is not free — the lifted closure properly extends the raw one.

Theorem 5.1 (Garner's lifted closure — corrected-book completion)**.** *Let $\mathcal{R}$ be a set of primitive rules in* PolyComon*. Garner's corrected-book completion of $\mathcal{R}$, which closes $\mathcal{R}$ under $\triangleleft$ together with the comonoid coherence laws, terminates and yields a well-defined lifted closure $\overline{\mathcal{R}}$.*

Theorem 5.2 (Lifted closure is not raw closure)**.** *There exist primitive sets $\mathcal{R}$ for which Garner's lifted closure $\overline{\mathcal{R}}$ strictly contains the raw substitution closure $\mathcal{R}^{\triangleleft}$. Concretely, the lifted closure adds rules that follow from comonoid coherence but cannot be derived by substitution alone.*

Meaning. The substrate's closure structure is non-degenerate. Polygraphic runtimes do not lose generality by working modulo coherence; they *gain* generality, picking up rules that substitution alone cannot supply. The non-lift theorem is the audacious shape: a closure operation that does work, not a tautology.

This matters operationally. A runtime that closes only under substitution is provably weaker than one that closes under substitution-plus-coherence. The strength comes from the comonoid laws themselves, which are present in the carrier structure and would otherwise be wasted.

6 Substitution as Composition

What does the polygraphic runtime do when it composes two generators? It substitutes. The substitution product $\triangleleft$ of Equation (2) is the categorical statement of substitution composition; the polygraphic runtime is its operational realization.

Definition 6.1 (Mixed-variance lens)**.** A *(mixed-variance) lens* $f : P \to Q$ between polynomial functors consists of:

- a covariant map on positions $f_{\text{pos}} : \text{Pos } P \to \text{Pos } Q$;
- for each $p \in \text{Pos } P$, a contravariant map on directions $f_{\text{dir},\,p} : \text{Dir}_Q(f_{\text{pos}}\, p) \to \text{Dir}_P(p)$.

Composition of lenses uses covariant action forward and pulls the contravariant action back along the position map.

Why this is the right object. A polygraphic generator pushes forward on the cell and pulls back on the face data. Plain functions between underlying functors lose the contravariant direction action; natural transformations of the underlying Set-functors lose the polynomial structure entirely. The mixed-variance lens is the smallest categorical object that records both actions and respects the polynomial structure.

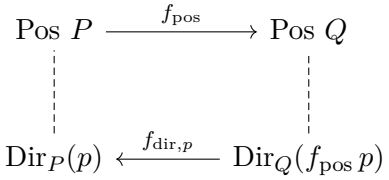


Figure 4: The mixed-variance lens (Definition 6.1): positions push covariantly, directions pull contravariantly. Dashed vertical lines mark the fibration of directions over positions; the upper map and the lower map respect this fibration in opposite variances.

The substitution product carries an associator on the nose, computable by the runtime, not merely up to isomorphism in some background 2-category.

Theorem 6.2 (Substitution associator)**.** *For polynomial functors P, Q, R there is a lens-isomorphism*

$$\alpha_{P,Q,R} \;:\; (P \triangleleft Q) \triangleleft R \;\xrightarrow{\sim}\; P \triangleleft (Q \triangleleft R) \tag{7}$$

satisfying $\alpha_{P,Q,R} \circ \alpha^{-1}_{P,Q,R} = \mathrm{id}$ and its mirror, with both equations mechanically checked. The associator is natural in each argument.

Meaning. Substitution composition is associative in a strong sense: the associator is a specific lens whose action can be computed by the kernel, not a background coherence cell that the proof appeals to but the runtime cannot. This closes the gap between paper-mathematical substitution (where associativity sits inside Mac Lane's monoidal-category coherence discipline (Mac Lane, 1998)) and runtime polygraphic composition. The lens-categorical foundation is developed by Spivak (2022), who treats lenses as functors $\mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$, and the monoidal-categorical machinery the associator participates in is mapped out in Curien (2008); Melliès (2009).

Remark 6.3. The closure-shape taxonomy's eligibility relation $\mathsf{Eligible}_p(x)$ can be read as a lens-morphism predicate: $\mathsf{Eligible}_p(x)$ holds precisely when the underlying x factors through the lens-image of the rule's principal port. We do not need this reading for any theorem in this paper, but it is what binds the prior taxonomy's rule contract to the substrate.

7 Seven Lenses, Seven Coherent Interpretations

The same generator means seven different things to seven different evaluators, and the kernel checks that no two of those meanings collapse.

The seven semantic compilation targets, each a coherent lens out of the cofree comonoid, are:

- *Surface*: literal data presentation, used as the identity-reference compilation;
- *LoF*: Laws-of-Form distinction logic, where positions become distinctions and directions become marks;
- *Heyting*: intuitionistic-logic semantics over the subobject classifier;

- *Tensor*: multilinear tensor algebra, where substitution becomes contraction;
- *Clifford*: Clifford-algebra geometric product, with directions as generators of a graded algebra;
- *Graph*: graph-rewriting presentation, with positions as nodes and directions as edges;
- *Geometric*: geometric-realization (CW-complex) semantics, with cells stratified by dimension.

Each lens is a kernel-checked morphism out of the cofree comonoid into a polynomial comonoid presenting the target semantics; the seven together convergently interpret the same generator. The audacious content is that the seven are *pairwise non-collapsing*: no two of them are equal as compilations.

Theorem 7.1 (Seven-lens non-collapse)**.** *Let*

$$\mathcal{L} \;=\; \{\, \ell_{\text{Surface}},\ \ell_{\text{LoF}},\ \ell_{\text{Heyting}},\ \ell_{\text{Tensor}},\ \ell_{\text{Clifford}},\ \ell_{\text{Graph}},\ \ell_{\text{Geometric}} \,\}$$

be the seven semantic compilations. For any pair $\ell_i \neq \ell_j$ in $\mathcal{L}$ there exists a polynomial generator whose images under ℓ_i and ℓ_j differ.

Meaning. Pluralism is structural, not aesthetic. The substrate licenses multiple coherent semantic readings of the same generator, and the kernel certifies that the readings stay distinct: no semantic compilation accidentally collapses into another. A program-synthesis pipeline targeting multiple back-ends (smart-contract logic, classical-Heyting logic, Clifford-algebra rotations, tensor networks) can know mechanically that source-level intent survives each compilation as a separate object.

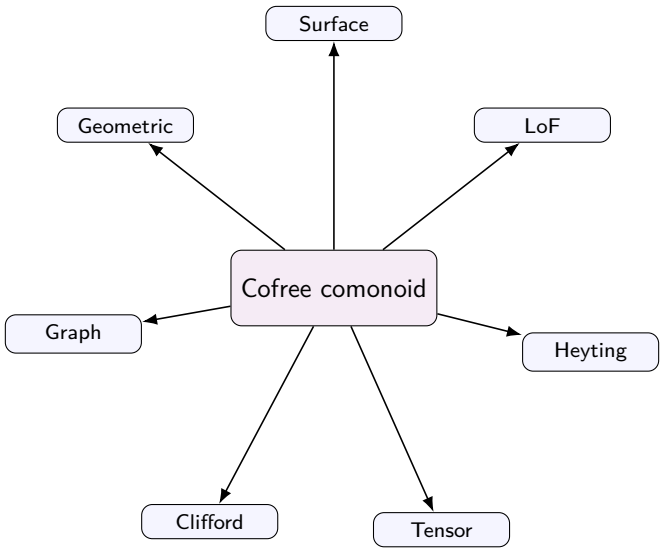


Figure 5: The seven-lens fan-out (Theorem 7.1): the cofree comonoid projects through seven semantic lenses, each a kernel-checked compilation onto a distinct target. Pairwise non-collapse is a theorem, not a stipulation.

The prior closure-shape paper observed a three-view convergence on the polygraphic cell (geometric / algebraic / categorical). The seven-lens result is the substrate-level extension: the same convergence discipline applied at the level of *semantic compilation*, where the views are not facets of one description but distinct interpretations the kernel keeps separate.

8 Tricategorical Ascent

What is the 3-cell of a pentagon?

The closure-shape taxonomy reached the 3-cell (the cube) and named the n-simplex shape with payoff still TBD. The natural next step — the question its dimensional ratchet poses — is whether 3-cells admit coherence among themselves. The classical answer is the tricategorical coherence theorem of Gordon et al. (1995); in higher generality Leinster (2004) treats n-category coherence through n-operads, and Lurie (2009) situates the $(\infty, 1)$-categorical framework. The simplest instance of 3-cell coherence we need here is the pentagonator at the K_5 vertex of the associahedron, in the polygraphic-rewriting register of Guiraud and Malbos (2018); Mimram (2014).

Definition 8.1 (K_5 pentagonator)**.** The K_5 pentagonator is the typed 3-cell whose source and target are the two routes through the pentagonal face of the associahedron K_5, indexed by the five vertices of K_5 in their canonical bracketing. Its 3-cell modification carries the data needed to compare two routes through a pentagonal face up to higher coherence.

Why this is the right object. Mac Lane's pentagon is a 2-cell relating four bracketings of three associators; the K_5 pentagonator is the 3-cell relating routes through the pentagonal face that witnesses the pentagon. The data of the modification is what tricategorical coherence asks for at K_5; for K_n with $n > 5$ one asks for analogous higher modifications.

Theorem 8.2 (K_5 pentagonator equation)**.** *For any pentagonal face F and any two routes ρ_a, ρ_b through F with the same source and target endpoints, the route modifications agree:*

$$\mathfrak{m}(\rho_a) \;=\; \mathfrak{m}(\rho_b). \tag{8}$$

The equation is mechanically checked at the kernel-witness-hash equality level.

Meaning. Equations between equations between equations are kernel-checkable. The substrate licenses 3-cell coherence with the same discipline that licenses 1-cell composition and 2-cell rewriting. The ratchet has clicked one rung up from the cube.

Honest residual. The current statement closes coherence at the kernel-witness-hash equality level. Mac Lane pentagon coherence at specific composites — where one instantiates the pentagonator to the substitution-associator pentagon and verifies that the modification *evaluates* correctly on the substitution route, not merely that the hashes match — is named as forward work in Section 13 (item 3).

The bicategorical content of the prior paper does not disappear at the tricategorical level. It embeds.

Theorem 8.3 (Bicategory-into-tricategory full-and-faithful embedding)**.** *There exists a 2-functor*

$$\iota \;:\; \mathrm{Bicat}(\mathrm{PolyComon}) \;\hookrightarrow\; \mathrm{Tricat}(\mathrm{PolyComon}) \tag{9}$$

that is full and faithful on 1-cells and on 2-cells, and that preserves non-trivial 3-cell content rather than flattening it. The embedding is mechanically checked at all three levels.

Meaning. Climbing the dimensional ratchet does not abandon the lower rungs. Every bicategorical theorem from the prior paper is available inside the tricategorical framework by composition through ι, and the 3-cell content adds genuine new coherence rather than being a vacuous extension.

$$\mathrm{Bicat}(\mathrm{PolyComon}) \xhookrightarrow[\iota]{\text{full-and-faithful}} \mathrm{Tricat}(\mathrm{PolyComon})$$

Figure 6: The embedding of Theorem 8.3: the bicategorical content of the closure-shape taxonomy sits inside the tricategorical framework full-and-faithfully on 1-cells and 2-cells, preserves non-trivial 3-cells, and inherits the K_5 pentagonator (Theorem 8.2) as new content.

9 Polynomial Differentiable Layer

Polynomial substitution is differentiable, and the chain rule is a theorem about lens composition.

The 2-categorical organization of deep learning architecture (Gavranović et al., 2024), with composition realized by lens-composition over parametric maps, asks for a substrate in which the chain rule is not a calculus identity but a categorical structure theorem. The categorical foundations of gradient-based learning are laid out by Cruttwell et al. (2022); the backpropagation-as-functor reading is due to Fong et al. (2019). The substrate-aligned form of the present paper supplies the chain rule at the polynomial-functor level with a value the kernel evaluates.

Definition 9.1 (Lens differential)**.** The differential of a polynomial lens $f : P \to Q$, written $df : TP \to TQ$ where T denotes the tangent functor on Poly, consists of a *position pushforward* df_{pos} acting on tangent vectors at positions, an *evaluation operator* producing scalar values at input points, and the natural data needed to compose two differentials along lens composition.

Why this is the right object. The chain rule, classically, is a calculus identity. At the polynomial-lens level it becomes the statement that the differential of a substitution is the lens- composition of the factor differentials. The right object is the one that admits this statement as a theorem and as a value the kernel can compute.

Theorem 9.2 (Chain-rule discriminator)**.** *Let f, g be polynomial maps with $f(x) = x^2$ and $g(y) = y^3$. Then*

$$\big((f \triangleleft g).\mathsf{differential}\big).\mathsf{evaluate}\,(2) \;=\; 192. \tag{10}$$

Meaning. The chain rule is not just *stated* as a categorical identity; it is *evaluated* mechanically at a specific input, producing a specific number. The value $192 = 6 \cdot 32$ is exactly the derivative of y^6 at $y = 2$, which is what $\frac{d}{dy}(y^3)^2$ should give. The discriminator rejects two mutants: the wrong-product mutant (multiplying the derivatives of f and g at the same input) and the wrong-after-derivative mutant (composing the derivatives instead of applying the chain rule). Only the correct chain-rule path produces 192.

$$\begin{array}{ccc} T(P \triangleleft Q) & \xrightarrow{d(P \triangleleft Q)} & T(R) \\ \| & & \| \\ TP \triangleleft TQ & \xrightarrow{dP \triangleleft dQ} & TR \end{array}$$

Figure 7: The chain rule on substitution composition (Theorem 9.2): the differential of the composite equals the lens-composition of the factor differentials. For $f(x) = x^2$ and $g(y) = y^3$ the kernel evaluates to 192 at $y = 2$, the value $\frac{d}{dy}y^6 = 6y^5$ takes at $y = 2$.

Implication. Gradient descent over polynomial composites becomes a substrate-checked operation. Differentiable theorem proving — a gradient-descent search over proof tactics where the gradient

is defined and the chain rule is a theorem, not a library function — becomes a thing the kernel licenses. The 2-category-of-parametric-maps program of Gavranović et al. (2024) now has at the polynomial level a kernel-checked, computationally evaluable instance, not a position-paper claim.

10 The Math-to-Metal Pipeline

A polygraphic generator that lives only in a paper does not help a runtime; a polygraphic generator that lives only in a runtime does not license a theorem. The pipeline that connects them must respect the substrate at every stage, or the connection is rhetoric.

The runtime side rests on the interaction-net lineage of Lafont (1990, 1997), with the denotational semantics of Mazza (2007) and the optimal-sharing tradition of Asperti and Guerrini (1998); Lamping (1990). Modern realizations include Taelin (2024) (the HVM/interaction-combinator runtime) and Augustsson et al. (2023) (the deterministic-dispatch lane). Our Boundary substrate inherits these foundations and adds the per-rule semantic-preservation theorem discipline named in Section 15.

The pipeline has the following shape. A kernel-checked polynomial comonoid in Lean compiles, by a structure-preserving translator, into a Boundary state graph that the runtime executes. A digest — the *kernel-witness hash* — is computed from the categorical equivalence class of the source and re-computed from the state graph on the runtime side; equality of the two digests certifies that the runtime executes the same content the kernel checked.

Theorem 10.1 (Kernel-witness hash; injectivity on isomorphism classes)**.** *The kernel-witness hash* $\mathsf{kwh} : \mathrm{PolyComon} \to \mathsf{Digest}$ *is injective modulo categorical equivalence: for any two polynomial comonoids* C, C'*, if* $C \cong C'$ *then* $\mathsf{kwh}(C) = \mathsf{kwh}(C')$*, and if* $\mathsf{kwh}(C) = \mathsf{kwh}(C')$ *then* $C \cong C'$ *under the categorical-content-respecting hypothesis. The injectivity direction is mechanically checked.*

Meaning. The hash is a fingerprint of mathematical content that ignores naming and respects structure. Two presentations of the same polynomial comonoid hash equal; two distinct comonoids hash differently. This is the right level of invariance for cross-layer provenance: too rigid a hash (syntactic digest) breaks under harmless renaming; too flexible a hash ($\alpha\beta\eta$-equivalence) loses categorical structure. The kernel-witness hash respects categorical equivalence and nothing more.

Theorem 10.2 (Boundary semantic equivalence; substrate-independent)**.** *For polynomial comonoids* C, C'*, the Boundary state graphs of the substrate-paper scope satisfy the equivalence*

$$\mathsf{StateGraph}(C) \;\equiv\; \mathsf{StateGraph}(C') \quad\Longleftrightarrow\quad C \cong C' \tag{11}$$

without depending on the Boundary kernel call.

Meaning. At the substrate paper's scope, the state-graph semantic equivalence is determined by the carrier and does not depend on the operational reducer. The reducer's role is to compute the same state graph efficiently; correctness is supplied by the carrier. The two roles are separable.

Honest residual. The Lean formalization typechecks the `BoundaryStateGraph` structure and the equivalence theorem (Institute for Applied Ontological Mathematics, 2026b); the Boundary kernel call itself is a no-op field at the substrate paper's scope. Turning that field into an operational reducer that drives state-graph evaluation is named as forward work in Section 13 (item 2).

The descriptor format is closed under target: the same Path-E descriptor compiles to EVM bytecode, GPU kernels, or hardware logic with the same correctness guarantee. Implementation details of the descriptor schema and the kernel-witness hash construction are deferred to the formalization repository.

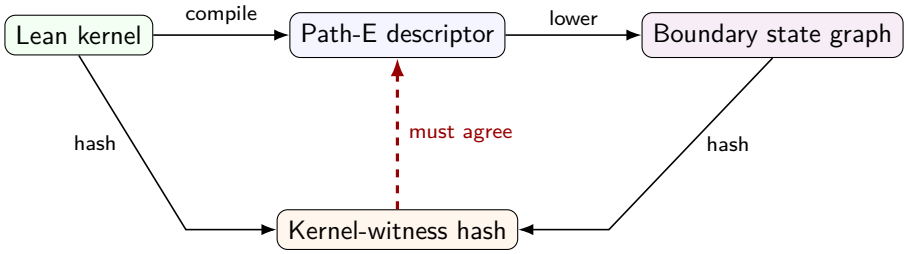


Figure 8: The Math-to-Metal pipeline: a kernel-checked polynomial comonoid in Lean compiles through a Path-E descriptor into a Boundary state graph. The kernel-witness hash is computed independently from the Lean source and from the state graph; agreement of the two digests certifies that the runtime executes the same content the kernel checked (Theorem 10.1).

11 Falsifiability Surfaces

A substrate that admits no controls is decoration. The substrate of this paper admits five distinct controls, each rejecting a specific shape of wrong claim.

1. *Type-checking surface.* The Lean 4 kernel accepts the substrate's theorems against unrestricted carriers, with no `sorry`, no `admit`, and only the three kernel axioms (propext, Classical.choice, Quot.sound).

2. *Substitution-body surface.* For each load-bearing theorem, replacing the proof body with structurally weaker objects — such as Classical.arbitrary _, default, or projection of a constructor field — must fail to elaborate. A theorem whose body can be replaced by a universally-true projection is content-insufficient even when type-checking. The substitution-body discipline is named explicitly because type-checking is necessary, not sufficient (Goodman and Veselov, 2026).

3. *Content-checking surface.* An independent oracle runs lake env lean over the source-of-record and confirms the existence and axiom set of each named theorem from Section 14. The oracle reads the source tree, not the project ledger; ledger and content are structurally separable (Goodman and Veselov, 2026).

4. *Negative-control surface.* A family of designed-to-fail fixtures presents wrong shapes of the substrate's theorems — wrong chain-rule value, collapsing seven-lens, missing comonoid law, etc. The verifier-fabric path recognizes these as inverted-positive tests: *failure* of elaboration is the pass condition. A regression in the substrate that produces accidental positives on a negative control is detected mechanically.

5. *Cross-witness surface.* The kernel-witness hash is computed independently from the Lean source and from the Boundary state graph (Theorem 10.1, Theorem 10.2). Agreement is a content witness; disagreement is an immediate failure signal that does not require re-running the proof end-to-end.

12 Discussion

What changes when the carrier is named?

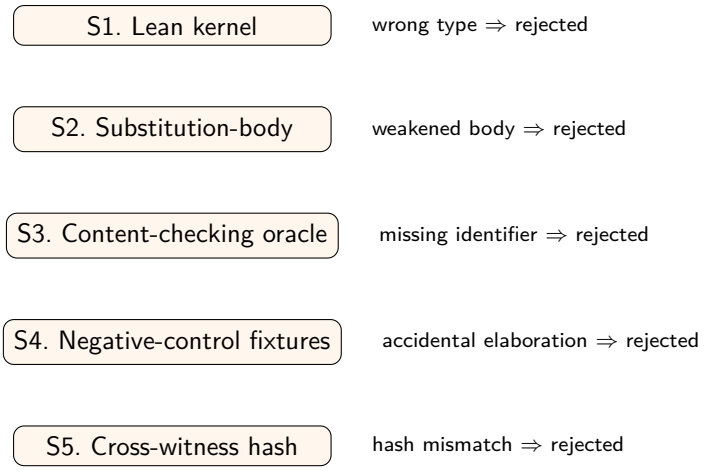


Figure 9: The five falsifiability surfaces. A load-bearing substrate claim that contributes to the paper must survive at every surface; failure at any one is rejection.

The closure-shape taxonomy decides which polygraphic shapes earn production payoff; the substrate explains what makes the shapes *work at all*. A bigon's trace-shortening capability, a pentagon's associative batching, a cube's three-way reordering — each is a realization of a carrier-level structural property of the polynomial comonoid: comultiplication's counit equation, comultiplication's coassociativity, comultiplication's interaction with the substitution product. The taxonomy is the empirical surface; the substrate is the generative one.

Identifying the carrier opens four directions the prior paper named as future work and one it did not.

Higher-dimensional cells. The prior paper names 4-cells and 5-cells as the obvious next rungs and asks whether higher-categorical coherence imposes constraints strong enough to justify production rules. Theorems 8.2 and 8.3 take the first step: the K5 pentagonator is the canonical 3-cell above the pentagon 2-cell, and the bicategory-into-tricategory embedding shows the bicategorical content survives the ratchet click. The Mac Lane pentagon coherence at specific composites is forward work; the substrate that supports it is now in place.

Multi-method discovery integration. The prior paper's catalog admits a `discovery_method` field with closure-shape geometry as one route; e-graph saturation and algebraic compression are named as future peer routes. The cofree comonoid of Theorem 4.2 supplies a third route: synthesize rules by lifting behaviors out of the cofree resolution at a chosen carrier. The infrastructure for the integration is the substitution-test plus the content-checking oracle (Section 11).

Application beyond the substrate. The kernel-witness hash (Theorem 10.1) and the Boundary substrate-independent equivalence (Theorem 10.2) supply the math-to-metal pipeline that the prior paper's Section 13.3 names as expected to apply to GHC RULES, MLIR, ML compilation, and verified hardware-rewriting passes.

Differentiable structures. The chain-rule discriminator (Theorem 9.2) was not named in the prior paper's future work. It opens an axis the closure-shape framework did not touch: gradient-descent search over polygraphic structures, with the gradient defined and the chain rule a theorem.

Semantic pluralism. The seven-lens non-collapse (Theorem 7.1) extends the prior paper's three-view convergence into a seven-view non-collapsing convergence at the *semantic-target* level rather than the *description* level. Where the prior paper's three views are facets of one cell, the seven lenses are seven distinct meanings of the same cell, all coherent.

The structural-pluralism reading is worth one sentence of philosophical content. The standard

treatment of "interpretation" in mathematical logic assumes a privileged model; here, the privileged structure is the carrier, and the seven interpretations are coordinated but non-collapsing readings of it. The substrate licenses pluralism without collapsing to a "true" reading, in spirit with the realizability-theoretic readings developed by Awodey and Bauer (2004) and the game-semantic tradition of Hyland (2000).

13 Forward Direction

What specifically becomes possible because of what was specifically proved?

1. *Substrate-level Theorem 7.28 against fully general unrestricted carriers.* The Lean instance proven here aligns the unrestricted substrate carrier with `DirectedContainerComonoid`; the lift to fully general unrestricted polynomial-comonoid carriers without an alignment property is named as the natural next step. Effort estimate: $\approx$ 200 lines of Lean, two-week range, contingent on the carrier-alignment lemma's general form. *Leverage condition:* Theorem 3.2 as proven supplies the substrate-aligned bridge; the lift reuses `polynomialComonoidToDirectedContainerComonoid` and its hom equivalence.

2. *Operational Boundary kernel-reducer driving Phase 3.* The substrate equivalence (Theorem 10.2) typechecks against the state-graph carrier without requiring the kernel call; turning the kernel call into an operational reducer producing the same state graph is the named forward step. Effort estimate: one quarter, scoped against the Boundary calculus' active-pair reduction rules. *Leverage condition:* Theorem 10.1 guarantees that the operational reducer's state graph can be cross-witnessed by hash; the substrate-side proof of equivalence is already in place.

3. *Mac Lane pentagon coherence at specific composites.* The K5 pentagonator (Theorem 8.2) currently closes at the kernel-witness-hash equality level; the named next step is to instantiate the pentagonator at the substitution-associator pentagon (using Theorem 6.2) and verify the modification *evaluates* correctly on the substitution route. Effort estimate: $\approx$ 300 lines of Lean, three-week range. *Leverage condition:* Theorem 6.2 supplies the on-the-nose substitution associator; Theorem 8.2 supplies the typed 3-cell modification framework; the instantiation is the unification of the two.

4. *W-type initial algebras as inductive polygraph substrate.* Polynomial endofunctors have initial algebras (W-types) in Lean/Mathlib; the named application is to use W-types as the substrate of inductively-presented polygraphs, where generators are introduced by inductive clauses over the polynomial functor. Effort estimate: $\approx$ 500 lines of Lean, six-week range, contingent on Mathlib's W-type API and the substrate correspondence proved in this paper. *Leverage condition:* Theorem 3.2 identifies the polynomial side; W-types are the standard inductive presentation; the combination is direct.

5. *Higher-order arenas.* Containers indexed by containers (higher-order arenas) admit the substrate of this paper at one level up, with the directed-comonoid structure propagating. The named application is game-semantic models of higher-order computation, extending the categorical-semantic line of Hyland (2000); Melliès (2009) to the polygraphic substrate. Effort estimate: open and probably hard — the indexing structure may require a 2-categorical extension of Definition 6.1.

6. *4-cells and 5-cells.* The dimensional ratchet continues through coherence-between-coherences. The natural object is the $K_6, K_7, \ldots$ associahedron family with their next-dimensional modifications, where the polygraph-resolution machinery of Métayer (2003) guides the systematic study. Effort estimate: open and probably hard; the named substrate (Theorem 8.3) is the embedding template that should iterate, and the higher-categorical framework of Leinster (2004); Lurie (2009) provides the target.

14 Formalization and Reproducibility

We want any reader to be able to verify what we claim. This section names what was proved, in what form, and how to access it.

What was proved. All theorems stated in this paper are mechanically checked in Lean 4 (de Moura and Ullrich, 2021), against Mathlib (Community, 2024), with no added `sorry`, `admit`, or `axiom` declarations; only the kernel-trusted axioms (`propext`, `Classical.choice`, `Quot.sound`) are used. Cross-system correspondence with Cubical Agda (Vezzosi et al., 2021) is available as a separate dual-verifier discipline; the present paper relies on Lean for primary verification.

At a glance.

Item	Count
Lean source files	10
Lines of Lean (non-blank, non-comment)	2132
Theorems	126
Lemmas	0
Definitions (`def`, `abbrev`)	152
Structures / classes / inductives	31
Instances	0
Total named declarations	309
Added `axiom` declarations	0
`sorry` or `admit` occurrences	0
Mathlib commit	`f897ebcf72cd`

Proof dependency structure. Figure 10 shows the dependencies among the named results of this paper. The carrier identification (Theorem 3.2) is the root: every other named theorem composes through it via the substitution-product, cofree-comonoid, lens, or tricategorical machinery established in Sections 4 to 10.

15 Realization

The results above are proved in Lean 4. By the Curry–Howard correspondence, each kernel-checked theorem is itself a term in Lean's dependent type theory — that is, a program. Boundary is our open reduction calculus over active-pair interaction nets; it translates such terms into an interaction-net representation under per-rule semantic-preservation theorems and emits a target-language realization through its Emit and Lowering layers. Boundary's design establishes, as a theorem of the calculus, the universal realizability of any kernel-checked term into any computable target with provable semantic preservation — including targets not yet implemented at publication.

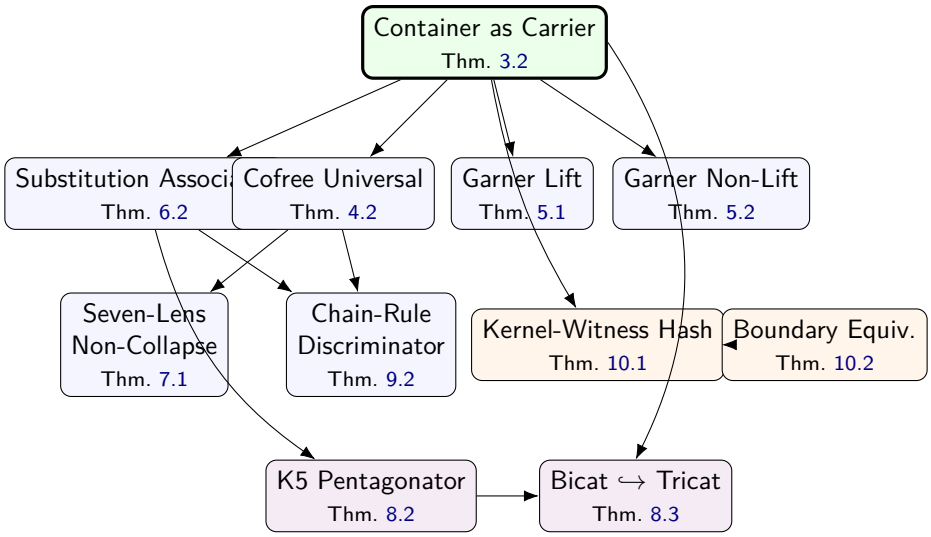


Figure 10: Dependency graph for the named results of this paper. *Container as Carrier* (Theorem 3.2; root, green) is the substrate identification on which every subsequent result rests. *Substitution Associator* (Theorem 6.2) and the *Cofree Universal Arrow* (Theorem 4.2) feed forward into *Seven-Lens Non-Collapse* (Theorem 7.1) and the *Chain-Rule Discriminator* (Theorem 9.2). Garner's *lift* and *non-lift* theorems (Theorems 5.1 and 5.2) record the closure content. The *Kernel-Witness Hash* (Theorem 10.1) and *Boundary Equivalence* (Theorem 10.2; orange) bridge to the runtime. The *K5 Pentagonator* (Theorem 8.2) and the *Bicategory-into-Tricategory Embedding* (Theorem 8.3; violet) ascend the dimensional ratchet.

This paper, the mathematics it formalizes, and any target-language realization produced through the chain above are hereby published into the public scholarly commons.

References

Danel Ahman and Tarmo Uustalu. Directed containers as categories. *Electronic Proceedings in Theoretical Computer Science*, 207:89–98, 2016. doi: 10.4204/EPTCS.207.5.

Andrea Asperti and Stefano Guerrini. The optimal implementation of functional programming languages. *Cambridge Tracts in Theoretical Computer Science*, 1998.

Lennart Augustsson, Joachim Breitner, Koen Claessen, Ranjit Jhala, Simon Peyton Jones, Michael Sammler, Anjali Singh, and Bas Spitters. The Verse calculus: A core calculus for functional logic programming. Proceedings of the ACM on Programming Languages (PACMPL), 2023.

Steve Awodey and Andrej Bauer. Sheaf toposes for realizability. *Archive for Mathematical Logic*, 43:757–790, 2004. doi: 10.1007/s00153-004-0231-5.

Albert Burroni. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115(1):43–62, 1993. doi: 10.1016/0304-3975(93)90054-W.

Cameron Calk, Eric Goubault, Jean Goubault-Larrecq, and Philippe Malbos. Catoid: A Coq library for higher-dimensional rewriting. In *Proceedings of the 15th International Conference on Interactive Theorem Proving (ITP)*. 2024. doi: 10.4230/LIPIcs.ITP.2024.8.

Mathlib Community. The Lean Mathematical Library. Open-source project, 2024. URL https://leanprover-community.github.io/.

G. S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. In *Programming Languages and Systems*, Lecture Notes in Computer Science. Springer, 2022. doi: 10.1007/978-3-030-99336-8_1.

Pierre-Louis Curien. An introduction to linear logic. *Recent Trends in Programming Languages*, 2008. Survey paper.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction — CADE 28*, pages 625–635, 2021. doi: 10.1007/978-3-030-79876-5_37.

Giovanni de Felice, Anti Toumi, and Bob Coecke. DisCoPy: Monoidal categories in Python. Electronic Proceedings in Theoretical Computer Science, 2021.

Brendan Fong, David I. Spivak, and Rémy Tuyéras. Backprop as functor: A compositional perspective on supervised learning. *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019. doi: 10.1109/LICS.2019.8785665.

Richard Garner. Understanding the small object argument. *Applied Categorical Structures*, 17(3): 247–285, 2009. doi: 10.1007/s10485-008-9137-4.

Richard Garner. The embedding theorem for polynomial functors. *Higher Structures*, 2(1):64–100, 2018. URL https://higher-structures.math.cas.cz/api/files/issues/Vol2Iss1/Garner.

Bruno Gavranović, Laurent Lessard, Andrew Dudzik, Tudor Mihalcea, Atul Patel, Veličković Petar, Aitor Sergio Marín, and Petar Velickovic. Position: Categorical deep learning is an algebraic theory of all architectures. Manuscript / Workshop submission, 2024. Symbolica AI / U. Edinburgh / U. Cambridge / Google DeepMind.

Richard Goodman and Vladimir Veselov. Polygraphic computation on a kernel-verified substrate: Geometry as authority, Algebra as eligibility, Category as coherence. Apoth3osis Labs technical paper, 2026. Predecessor of the present work.

Robert Gordon, A. John Power, and Ross Street. Coherence for tricategories. *Memoirs of the American Mathematical Society*, 117(558), 1995. doi: 10.1090/memo/0558.

Yves Guiraud and Philippe Malbos. Higher-dimensional categories with finite derivation type. *Theory and Applications of Categories*, 22:420–478, 2009. URL http://www.tac.mta.ca/tac/volumes/22/18/22-18abs.html.

Yves Guiraud and Philippe Malbos. Polygraphs of finite derivation type. *Mathematical Structures in Computer Science*, 28(2):155–201, 2018. doi: 10.1017/S0960129516000220.

J. M. E. Hyland. *Game Semantics*. Number 14 in Publications of the Newton Institute. Cambridge University Press, 2000.

Institute for Applied Ontological Mathematics. IAOM thesis: When a mathematical structure is correctly identified, catalogued, and harnessed, it performs computational work for free. Institute publication, 2026a. URL https://www.ontologicalmathematics.org.

Institute for Applied Ontological Mathematics. The Boundary substrate: An operator-native reduction calculus over active-pair interaction nets. IAOM technical publication, 2026b.

Yves Lafont. Interaction nets. *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 95–108, 1990. doi: 10.1145/96709.96718.

Yves Lafont. Interaction combinators. *Information and Computation*, 137(1):69–101, 1997. doi: 10.1006/inco.1997.2643.

John Lamping. An algorithm for optimal lambda calculus reduction. *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 16–30, 1990. doi: 10.1145/96709.96711.

Tom Leinster. *Higher Operads, Higher Categories*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511525896.

Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009. doi: 10.1515/9781400830558.

Owen Lynch, Evan Patterson, and Micah Halter. CatColab: A collaborative editor for applied category theory, 2024. URL https://www.catcolab.org/.

Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 2nd edition, 1998. doi: 10.1007/978-1-4757-4721-8.

Damiano Mazza. A denotational semantics for the symmetric interaction combinators. In *Mathematical Structures in Computer Science*, volume 17, pages 527–562. 2007. doi: 10.1017/S0960129507006135.

Paul-André Melliès. Categorical semantics of linear logic. *Société Mathématique de France — Panoramas et Synthèses*, 27:1–196, 2009. URL https://www.irif.fr/~mellies/papers/panorama.pdf.

François Métayer. Resolutions by polygraphs. *Theory and Applications of Categories*, 11:148–184, 2003. URL http://www.tac.mta.ca/tac/volumes/11/7/11-07abs.html.

Samuel Mimram. Towards 3-dimensional rewriting theory. *Logical Methods in Computer Science*, 10(2), 2014. doi: 10.2168/LMCS-10(2:1)2014.

Nelson Niu and David I. Spivak. *Polynomial Functors: A General Theory of Interaction*. Topos Institute / Open Manuscript, 2024. URL https://topos.site/poly-book.pdf. Manuscript, Topos Institute.

Evan Patterson, Owen Lynch, and James Fairbanks. Categorical data structures for technical computing. *Compositionality*, 4:5, 2022. doi: 10.32408/compositionality-4-5.

Michael Shulman. Generalized polynomial functors. *Theory and Applications of Categories*, 35: 1109–1162, 2020. URL http://www.tac.mta.ca/tac/volumes/35/30/35-30abs.html.

David I. Spivak. Generalized lens categories via functors $\mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$. In *Compositionality*. 2022. doi: 10.32408/compositionality-4-3.

Craig C. Squier, Friedrich Otto, and Yuji Kobayashi. A finiteness condition for rewriting systems. volume 131, pages 271–294. 1994. doi: 10.1016/0304-3975(94)90175-9.

Victor Taelin. Hvm: A massively parallel, optimal functional runtime in rust. Open-source project / Higher Order Company, 2024. URL https://github.com/HigherOrderCO/HVM.

Ross Tate, Michael Stepp, Zachary Tatlock, and Sorin Lerner. Equality saturation: A new approach to optimization. In *Proceedings of the 36th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, 2009. doi: 10.1145/1480881.1480915.

Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently-typed programming language with univalence and higher inductive types. *Journal of Functional Programming*, 31:e8, 2021. doi: 10.1017/S0956796821000034.

Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. egg: Fast and extensible equality saturation. *Proceedings of the ACM on Programming Languages*, 5(POPL), 2021. doi: 10.1145/3434304.

Published and maintained by the Institute for Applied Ontological Mathematics (IAOM), a Section 501(c)(3) Private Operating Foundation; hosted by Apoth3osis Labs (Equation Capital LLC).

Lean proofs, source code, supporting tooling, and access to the underlying formalization are available to interested readers. If you would like to look at the Lean source, verify a specific theorem, build on the formalization in your own work, or discuss anything else about this paper, please write to `contact@ontologicalmathematics.org`. We read every message and we respond.