Geometric Computation: Twin Helices and the Algebra of Flow

Benjamin Kaminsky*

September 18, 2025

Abstract

This work constructs a manifold consisting of twin helices to investigate the existence of computational geometric bordisms, as part of the broader program of the cobordism hypothesis. We study its geometric, topological, algebraic, and computational properties, suggesting the emergence of a new research direction in categorical physics termed Geometric Computability Theory, situated at the intersection of geometry, topology, and computability theory. We show that these "Geometrically Computable Manifolds" can encode arbitrary cyclic groups, leading to universal computational behavior in the sense of Kleene's partial recursive functions. We further analyze generalized n-ary computational dynamics and formulate a metatheorem describing trade-offs inherent in the computational regimes. Ordinary differential equations are used to model discrete fluidic computation, linking the construction to continuous dynamics. This perspective opens avenues for future investigation, including potential applications to Yang–Mills theory. As of writing, no algebraic systems yet exist to formalize the behavior of these manifolds; future work will develop such frameworks along with rigorous computational models and experimental simulations.

Keywords: geometric computation, dynamical systems, prequantum geometry, Yang-Mills theory, cobordism hypothesis, helices, computational gauge theory, topological quantum field theory.

*benkaminskymath@gmail.com

Contents

Introduction

"A twisted version of four-dimensional supersymmetric gauge theory is formulated..."

— Edward Witten, *Topological Quantum Field Theory* (1988)

This work presents a geometric model of computation constructed from helically structured manifolds, with morphisms defined by flow transformations that preserve finite-energy symbolic invariants. The resulting category is enriched over a symmetric monoidal category of structured flows—including chirality, join angle, and torsion data—enabling a deterministic yet expressive semantics that encodes symbolic computation directly in geometric topology.

Section 1 begins with the construction of a deterministic geometry formed from manifolds made of helices and, in doing so, lays out the coherence conditions based on the gluing constraints imposed by chirality and torsion. From this construction arises a new class of computational objects: symbolic flow manifolds that admit non-cancellative monoidal structure. From there on, we execute a series of calculations for the group-theoretic and geometric/topological behavior of this class of manifolds.

Section 2 sets out a framework for a theory of physical computation. The creation of a definition for a "Geometrically Computable Manifold" and its resulting state vector, the "Geobit." The formulas for geometric quantization are established. Here we attempt to reconcile semantic expressiveness in programming languages (in the sense of Felleisen and Sabry) with the structural stability and reversibility of geometric processes. While syntactic semantics often struggles to formalize languages with rich dynamic structure, the approach taken here replaces syntactic encoding with differential-geometric ones, specifically, moduli spaces of helices and their categorical behavior under symbolic flow. The resulting enriched category naturally supports symbolic computation with bounded state and fixed-point behavior, and exhibits expressive capacity that surpasses classical Von Neumann-like models (e.g., Turing machines and register machines) while retaining verifiability at the topological level. By analyzing this relationship, we arrive at the central result of the paper—a conservation law for computational behavior itself.

Section 3 grounds this resulting framework rigorously in the theory of Kleene's partial recursive functions, and in doing so, provides a natural transition point from finite groups to groupoids, where computation becomes locally sensitive and context-dependent.

Section 4 describes the algebraic behavior of these manifolds as the computation evolves towards increasingly fluid-like behavior under the groupoid framework. A new counterpart to the finite state machine is introduced as an alternative to the use of global PDEs. Proposals are made for a future method of modeling of discrete fluidic computation via networks of helix-based Lie groupoids and an extended homotopy-theoretic framework to support such a model. A geometric counterpart to measuring Felleisen-style macro-expressiveness in chaotic analog processes is proven via harmonic analysis. Research direction towards the future of algebraic frameworks and a fully constructive classification of computational behavior inside TQFTs is proposed.

Section 5 situates our model within a larger categorical context. The beginnings of a "Computational Cobordism Hypothesis" are laid out in addition to formalization and comparison of Post-Turing complete forms of computation in the context of Prof. Lurie's higher algebra. Following this, we apply our model to the foundations of physics itself, attempting to reveal the origin of computational randomness as a quotient of geometric complexity, a coarse-graining of torsion and chirality across moduli spaces. This gives a concrete realization of what may underlie quantum indeterminacy: a deterministic symbolic process whose full structure is not observable from any single projection. Further research directions are formulated towards concrete applications and relations to both string and twistor theory. We conclude with a conjecture—inspired by the Lost Melody theorem of Hamkins and Lewis—that formalizes this idea as a boundary condition on computable real analysis and symbolic flow.

Despite the peril, we remain hopeful that a new form constructive computational gauge theory based on the philosophy of Alexander Grothendieck may offer a new way forward in the study of HEP.

This paper is, by its nature, an experiment in synthesis as much as in rigor. If its scope sometimes appears unruly, or its exposition occasionally ventures beyond traditional boundaries, it is only because the questions at hand resist treatment by standard means. If you are to read it, I ask you to do so with care and without feeling pressured to do so all at once. I have aimed for clarity without sacrificing depth, so if the arguments sometimes feel unconventional, I hope that patient reading and engagement will reward the effort.

Ultimately, my intent is not to assert authority, but to invite the reader to explore together an unexpected landscape that may, in time, reshape our understanding of computation itself.

1 Twin Helical Manifolds

1.1 Helical Solids

We construct a helical solid by the following method. Begin with the standard parametric equation for an idealized helix. Let the general parametric form for our helix $\gamma(\theta)$ be:

$$\gamma(\theta) = (R\cos(\theta), R\sin(\theta), h\theta)$$

We now define the solid points of the helical object using the following equation:

$$S = \{\mathbf{S}(\theta, r', \phi) \mid \theta_{\min} \leq \theta \leq \theta_{\max},\ 0 \leq r' \leq \rho(\theta),\ 0 \leq \phi < 2\pi\}$$

Here, $\mathbf{S}(\theta, r', \phi)$ denotes the point in $\mathbb{R}^3$ obtained by starting at position $\gamma(\theta)$ on the helical spine, then moving a distance r' in the plane normal to the spine at angle ϕ. Let $\rho(\theta)$ define the cross-sectional radius of the helix for θ in some domain $[\theta_{\min}, \theta_{\max}]$.

Definition 1.1 (Helix Base and Apex)**.** The *base* of the helix [1] is defined as the point where $\rho(\theta)$ attains its maximal value:

$$\rho_{\text{base}} := \rho(\theta_{\min})$$

The *apex* is defined as the point where $\rho(\theta)$ attains its minimal value $\theta = \theta_{\max}$:

$$\rho_{\text{apex}} := \rho(\theta_{\max})$$

Let $\rho(\theta)$ be:

$$\rho(\theta) \;=\; \rho_{\text{apex}} \;+\; \big(\rho_{\text{base}} - \rho_{\text{apex}}\big)\ \underbrace{\cos^2\!\big(\tfrac{\pi}{2}\,u\big)}_{f(u)}$$

Where:

$$u = \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}$$

and $0 < \rho_{\text{apex}} < \rho_{\text{base}}$. We select a $\cos^2$ profile, as it has vanishing derivatives at both ends. Next, to eliminate any possibility for a singular tip, we replace the terminal disk at $\theta = \theta_{\max}$ with a smooth, dome-shaped manifold patch. This patch, which we term the apical cap, is constructed as a surface of revolution that guarantees a C^1 join with the main body of the helical solid. Let (T, N, B) be the Frenet-Serret frame of the helical spine $\gamma(\theta)$ evaluated at the apex, $\theta = \theta_{\max}$. The construction of the cap is as follows.

Definition 1.2 (Smooth Apical Cap)**.** The apical cap, $S_{\text{cap}}(u, v)$, is a surface parameterized for $u \in [0, \pi/2]$ and $v \in [0, 2\pi)$ by the equation:

$$S_{\text{cap}}(u, v) = \gamma(\theta_{\max}) + \rho_{\text{apex}}\sin(u)T + \rho_{\text{apex}}\cos(u)(\cos(v)N + \sin(v)B)$$

where $\rho_{\text{apex}} = \rho(\theta_{\max})$ is the minimal radius of the helical tube.

Remark 1.1. The boundary of this cap at $u = 0$ corresponds to the terminal circle of the main helical tube. We verify that the tangent planes of the cap and the tube are identical at this junction. The partial derivatives of the cap's parameterization at the junction ($u = 0$) are:

$$\left.\frac{\partial \mathbf{S}_{\text{cap}}}{\partial u}\right|_{u=0} = \rho_{\text{apex}}\mathbf{T}$$
$$\left.\frac{\partial \mathbf{S}_{\text{cap}}}{\partial v}\right|_{u=0} = \rho_{\text{apex}}\left(-\sin(v)N + \cos(v)B\right)$$

The tangent plane of the cap at the junction is spanned by these two vectors. The first vector, $\left.\frac{\partial \mathbf{S}_{\text{cap}}}{\partial u}\right|_{u=0}$, is parallel to the helical spine's tangent vector T at the junction. The second is tangent to the circular cross-section. This matches the tangent plane of the main helical body at $\theta = \theta_{\max}$, where the condition $\rho'(\theta_{\max}) = 0$ ensures its longitudinal tangent is also parallel to T. Therefore, the apical cap joins the main body seamlessly, and the resulting solid possesses a C^1 manifold boundary.

[1] There is not, in fact, an accepted term for this, so we adapt one from conics

Remark 1.2. Let $0 < \rho_{\text{apex}} < \rho_{\text{base}}$ to avoid a cone point. We impose $\rho_{\text{base}} < R$ and $\rho_{\text{base}} < \pi h$ so that the inflated tube is embedded; no further restriction on $\theta_{\min}$ is required.

1.2 Chirality and Alignment

Definition 1.3 (Chirality)**.** A subset $O \subset \mathbb{R}^n$ is said to have *chirality* if it is not properly congruent to its mirror image. Equivalently, O is chiral if for every reflection r across a plane in $\mathbb{R}^n$, there exists no proper isometry, i.e. orientation-preserving Euclidean motion (translations or rotations), g such that $g(O) = r(O)$. We say that helices may possess either left-handed or right-handed chirality. A helix is left-handed if $h < 0$ and right-handed if $h > 0$.

Definition 1.4 (Reference Frame)**.** Throughout this work, we fix the standard right-handed coordinate frame in $\mathbb{R}^3$. The positive z-axis, $\vec{e}_z$, serves as the primary axis of reference for defining chirality and orientation. The reference hyperplane is the xy-plane at $z = 0$. All geometric, group, and logical actions are measured relative to this frame.

Lemma 1.1 (Spiral Matching Lemma)**.** *Two Archimedean spirals, S_1 and S_2, with distinct centers and the same pitch, can intersect to form a C^0 curve with a matching phase if and only if they have opposite chirality.*

Proof. Let the two spirals be defined in $\mathbb{R}^2$ by the vector equations:

$$S_i(\theta_i) = C_i + a_i\theta_i \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix}, \quad i = 1, 2$$

We are given that the centers are distinct ($C_1 \neq C_2$) and the pitches are equal ($|a_1| = |a_2| = a > 0$). The lemma requires two conditions to be met simultaneously:

(a) C^0 **Intersection** — The spirals meet at a common point. $S_1(\theta_1) = S_2(\theta_2)$

(b) **Phase Matching** — The local polar angles are the same. $\theta_1 = \theta_2 =: \theta$

Substitute the phase-matching condition into the intersection equation:

$$C_1 + a_1\theta \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} = C_2 + a_2\theta \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

Rearranging the terms to isolate the vector between the centers gives our primary relation:

$$C_2 - C_1 = (a_1 - a_2)\theta \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

We analyze this relation for both chirality cases.

(a) **Same Chirality** — ($a_1 = a_2$) If the spirals have the same chirality, the term $(a_1 - a_2)$ is zero. The relation becomes:

$$C_2 - C_1 = 0 \implies C_1 = C_2$$

This contradicts the given condition that the centers are distinct. Therefore, an intersection with matching phase is impossible for spirals of the same chirality.

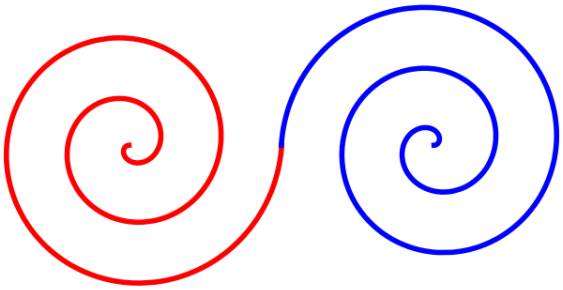
(b) **Opposite Chirality** — ($a_2 = -a_1$) If the spirals have opposite chirality, the term $(a_1 - a_2)$ becomes $(a_1 - (-a_1)) = 2a_1$. The relation becomes:

$$C_2 - C_1 = 2a_1\theta \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

Thus, an intersection with matching phase is possible if and only if the spirals have opposite chirality. □

Remark 1.3. We impose the phase-matching condition $\theta_1 = \theta_2$ so that the radial directions—and hence the unit tangent directions—agree at the intersection, removing any kink, the necessary first step toward any C^1 or C^2 splice.

Corollary 1.2 (Coaxial Spiral Chirality Constraint)**.** *Let S_1 and S_2 be Archimedean spirals centered at the same point. Then, a C^0 bounded path connecting them with matching phase at a common point can exist only if their chiralities are identical.*



Proof. This is a direct consequence of the Spiral Matching Lemma. As proven above, with $C_1 = C_2$, equation:

$$(C_2 - C_1) = (a_1 - a_2)\theta(\cos\theta, \sin\theta)$$

which gives:

$$(a_1 - a_2)\theta\,(\cos\theta, \sin\theta) = 0$$

for $\theta \neq 0$ this forces $a_1 = a_2$, which in turn requires the spirals to have the same chirality. □

Remark 1.4. This illustrates a fundamental principle in geometric design: adding a simple local constraint, like matching phase, can force a relationship between the global properties (chirality) of the curves. In this case, the seemingly mild phase-matching condition is powerful enough to forbid a connection entirely for same-chirality spirals with offset centers.

1.3 Twin Helical Manifolds

We now define a mirrored helix and perform a topological gluing operation to form a pair of helices with opposing chirality, joined at their respective bases.

Definition 1.5 (Binary Orientation of a Helix)**.** Let H be a helix in $\mathbb{R}^3$ with axis of symmetry parallel to the z-axis. The orientation of H is defined by the monotonicity of its z-coordinate as a function of its intrinsic parameter u, increasing from base to apex:

(a) Upwards-oriented — $z(u)$ is strictly increasing from base to apex.

(b) Downwards-oriented — $z(u)$ is strictly decreasing from base to apex.

Definition 1.6 (Types of Twin Helices)**.** To generate a Type 1 Twin Helix, we fix $R > 0$ and pitch $h > 0$. Let the first helical solid H_A be that which corresponds to the spine:

$$\gamma_A(\theta) \;=\; (\,R\cos\theta,\; R\sin\theta,\; h\theta\,), \quad \theta \in \mathbb{R}$$

Let the second helical solid H_B be one rotated by π about the x–axis to obtain the twin:

$$\gamma_B(\theta) = (\,R\cos\theta,\; -R\sin\theta,\; -h\theta\,)$$

- γ_B is right-handed if γ_A is left-handed, and vice-versa.
- At $\theta = 0$, the spines γ_A and γ_B both pass through the point $(R, 0, 0)$. The tangent vectors at this point, $\gamma_A'(0) = (0, R, h)$ and $\gamma_B'(0) = (0, -R, -h)$, have opposite y- and z-components, ensuring the bases can be joined smoothly.

To generate the remaining types, let $\mathcal{G}$ be the specific subgroup of $E(3)$ generated by the elements as outlined in the final column of 1. Applying appropriate elements of $\mathcal{G}$ to H_B produces the remaining three configurations as displayed in 1.

To apply the topological gluing, let H_A and H_B be compact 3-manifolds with boundary embedded in $\mathbb{R}^3$, representing the helical manifolds parameterized by γ_A, γ_B respectively. Each H_i $(i = A, B)$ has a single boundary component ∂H_i, which corresponds to the helix's base. Let $B_i \subset \partial H_i$ denote the base surface of H_i, corresponding to the endpoint 0 of the interval $[0, 1]$ in the product structure $D^2 \times [0, 1]$. Since we assume closed cross-sections, each B_i is homeomorphic to the closed 2-disk D^2.

The local diffeomorphism that creates the bridge is realized by a parametric curve, $\Gamma(s)$, constructed as a Cubic Hermite spline[1]. This curve provides the central spine of the connecting manifold.

Definition 1.7 (Base-to-Base Diffeomorphic Bridge)**.** The spine of the bridge, $\Gamma(s)$, for a parameter $s \in [0,1]$ is given by:

$$\Gamma(s) = h_{00}(s)P_A + h_{01}(s)P_B + h_{10}(s)\mathbf{m}_A + h_{11}(s)\mathbf{m}_B$$

where:

(a) P_A and P_B are the base positions.

(b) The tangent vectors $\mathbf{m}_A$ and $\mathbf{m}_B$ are defined by the unit tangents $\mathbf{t}_A, \mathbf{t}_B$, horizontal or vertical as required, and a scalar tension parameter ℓ:

$$\mathbf{m}_A = \ell\mathbf{t}_A \quad \text{and} \quad \mathbf{m}_B = \ell\mathbf{t}_B$$

(c) The functions $h_{ij}(s)$ are the standard cubic Hermite basis functions:

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1 \\ h_{01}(s) &= -2s^3 + 3s^2 \\ h_{10}(s) &= s^3 - 2s^2 + s \\ h_{11}(s) &= s^3 - s^2 \end{aligned}$$

For Types 2 and 4, we offset the neck by the bump function:

$$\widetilde{\Gamma}(s) = \Gamma(s) + \alpha s^2(1-s)^2\mathbf{n}$$

where:

(a) $\mathbf{n}$ is any unit vector perpendicular to both tangents (in the coaxial case, take the outward radial direction $(0,1,0)$)

(b) $\alpha > 2\rho_{\text{base}}$ guarantees the neck clears the parent tubes, which preserves C^1 boundary data while preventing the Hermite loop from intersecting the thickened tubes.

Let $\rho_{\text{bridge}}(s)$ be the radius of the bridge tube as a function of the spline parameter s. A simple choice is a constant radius $\rho_{\text{bridge}}(s) = \rho_{\text{base}}$. The points in the bridge are given by:

$$P(s, r', \phi) = \Gamma(s) + r'(\cos(\phi)N(s) + \sin(\phi)B(s))$$

for $0 \le r' \le \rho_{\text{base}}$, where $(T(s), N(s), B(s))$ is the Frenet-Serret frame of the bridge spine $\Gamma(s)$.

For the coaxial same-orientation case (Type 2), we take the bump in a horizontal direction $\mathbf{n} \perp \mathbf{t}_A$ rather than in $\mathbf{e}_z$, moving the neck off the common axis and preventing the Hermite hair-pin from self-intersecting.

By construction, this spline and bump function rigorously satisfies the boundary conditions $\Gamma(0) = P_A$, $\Gamma(1) = P_B$, $\Gamma'(0) = \mathbf{m}_A$, and $\Gamma'(1) = \mathbf{m}_B$ that joins the two helical solids at their bases with a minimum of C^1 continuity, in addition to correctly realizing the global features of the Type 2, 3, and 4 configurations.

H_A and H_B are the compact 3-manifolds with boundary representing the two helical solids. Their bases, $B_A \subset \partial H_A$ and $B_B \subset \partial H_B$, are the surfaces to be joined. We form the disjoint union $H_A \sqcup H_B$ and define an equivalence relation $\sim$ by identifying points on the bases via a diffeomorphism, $p \sim f(p)$ for $p \in B_A$. The resulting quotient space,

$$H_{AB} = (H_A \sqcup H_B)/\sim,$$

is the abstract topological manifold formed by this gluing.

To realize H_{AB} as a smooth (C^1) embedded manifold, we employ the explicit geometric bridge detailed previously. The Cubic Hermite spline provides a smooth transition between the bases of H_A and H_B, ensuring that the tangent planes match precisely at the junctions and have at least C^1 (and, with higher-degree splines, potentially $C^{2,3}$) regularity.

We term this new construction, a smooth manifold embedded in $\mathbb{R}^3$, a *Twin Helical Manifold* ("THM"), or equivalently, just a *Twin Helix*.

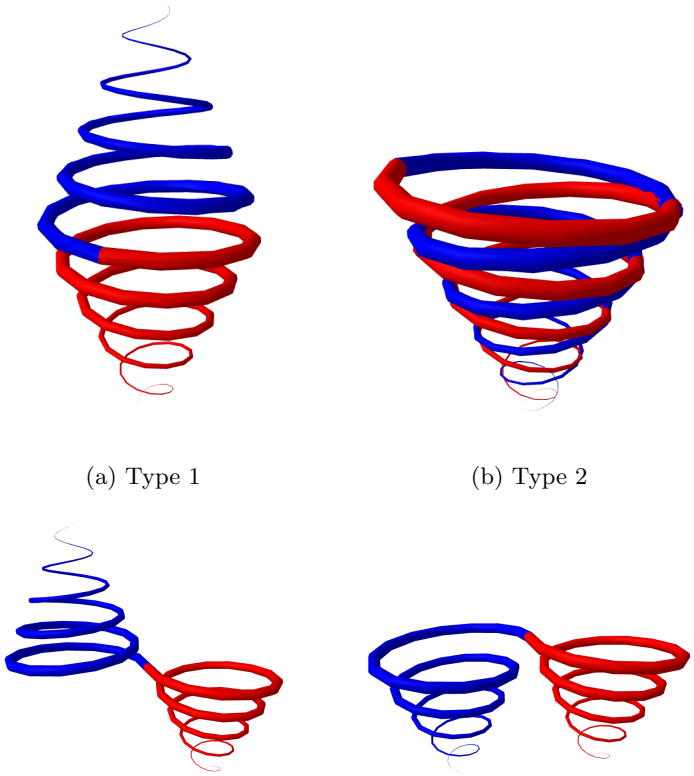
Definition 1.8 (Twin Helical Manifold and Unit Helices)**.** A THM is defined as a smooth manifold in $\mathbb{R}^n$ formed by a topological gluing of two helical solids and a diffeomorphic bridge that smoothly joins their respective base surfaces. The two helices composing a single THM are termed *unit helices*, denoted H_A and H_B.

1.4 The Group Structure and Geometric Topology of Twin Helices

By combining different chiralities and orientations, we define a geometric family of four canonical manifolds, denoted THM_4. This yields a total of 4 distinct invariant configuration states.

Table 1: Essential Type Chart for Twin Helical Manifolds

Type	**Chirality**	**Orientation**	**Axiality**	**Mnemonic**	**Map** $\Rightarrow H_B$
1	Opposite	Opposite	Coaxial	Mirrored Horn	$R_x(\pi)$
2	Opposite	Same	Coaxial	Twisting Pair/DNA-like	Σ_y
3	Same	Opposite	Offset	Mirrored Symmetrical Flow	$R_y(\pi), T$
4	Same	Same	Offset	Pair of Horns/Drills	$R_z(\pi), T$



(a) Type 1 (b) Type 2

(c) Type 3 (d) Type 4

Figure 1: The THM_4 Geometric Family.

However, our Type 2 helix has a small problem. A simple drawing reveals that constructing a twin helix with its attributes becomes problematic in the previously defined $\mathbb{R}^3$ space. A closer investigation leads to the following conclusion.

Lemma 1.3 (Twin Helix Intersection Lemma)**.** *No twin helical manifold whose unit helices with turns $N \geq 1/2$, sharing both axial alignment, orientation, and opposing chirality can exist without intersection of its respective unit helices in $\mathbb{R}^3$.*

Proof. Let H_A and H_B be two coaxial helices parameterized by $t \in [0, 1]$. Both helices share the same radial function $r(t)$ and the same axial function $z(t)$. Their distinction lies in chirality and a possible phase offset.

The angular position of a point on helix H_x at parameter t is given by:

$$\alpha_x(t) = 2\pi N \chi_x t + \phi_x,$$

where:

(a) N is the number of turns in one unit helix,

(b) χ_x is the chirality of H_x, with $\chi_A = -1$ and $\chi_B = 1$ for Type 2,

(c) ϕ_x is the phase offset. Without loss of generality, we may set $\phi_A = 0$.

The center lines of the two helices intersect if, for some $t \in [0, 1]$, their angular positions coincide modulo 2π. That is:

$$\alpha_B(t) - \alpha_A(t) = 2k\pi$$

for some integer k. Substituting the expressions for $\alpha_A(t)$ and $\alpha_B(t)$, we obtain:

$$\begin{aligned}(2\pi N\chi_B t + \phi_B) - (2\pi N\chi_A t) &= 2k\pi \\ 2\pi Nt(\chi_B - \chi_A) + \phi_B &= 2k\pi \\ 4\pi Nt + \phi_B &= 2k\pi.\end{aligned}$$

Define the function $f(t) = 4\pi Nt + \phi_B$. An intersection occurs if there exists a $t \in [0, 1]$ such that $f(t) = 2k\pi$ for some integer k.

The function $f(t)$ is continuous and monotonic on $[0, 1]$ (assuming $N > 0$). The values it takes range from:

$$f(0) = \phi_B, \quad f(1) = 4\pi N + \phi_B.$$

This interval has length $4\pi N$. For $f(t)$ to contain at least one multiple of 2π, it suffices that:

$$4\pi N \geq 2\pi \quad \Rightarrow \quad N \geq \frac{1}{2}.$$

By the Intermediate Value Theorem, since $f(t)$ is continuous on $[0, 1]$, it must attain all values in the interval $[\phi_B, \phi_B + 4\pi N]$. If this interval has length at least 2π, it must contain at least one value $2k\pi$. Therefore, for $N \geq \frac{1}{2}$, there always exists a $t \in [0, 1]$ for which $\alpha_B(t) \equiv \alpha_A(t) \mod 2\pi$, and thus the helices intersect.

Hence, for any $N \geq \frac{1}{2}$, it is mathematically impossible to choose a constant angular phase offset ϕ_B that avoids intersection of the center lines of the Type 2 twin helices on $[0, 1]$. □

Remark 1.5. Such is the nature of V_4. Like its close cousin, the Klein bottle, manifolds in this symmetry class resist consistent embedding in $\mathbb{R}^3$ without self-intersection. The issue arises from an implicit non-orientable twist in the configuration. It becomes analogous to the problem of embedding a Klein bottle in $\mathbb{R}^3$ without singularities. These twists violate the global orientability and parallelizability needed for a smooth immersion. While the resulting manifold is orientable, its specific geometric configuration creates an embedding problem analogous to that of non-orientable manifolds, where an additional dimension is required to resolve self-intersection.

This is a known result in algebraic topology [2], though its appearance here is unconventional. Rather than be discouraged, we now move into $\mathbb{R}^4$, where the additional dimension allows a clean, self-consistent embedding of these twist-preserving manifolds—preserving both the geometric and group-theoretic structure of the system.

Remark 1.6 (Chirality Rule-Reversal Between 2- and 3-Dimensional Gluing). A curious but systematic reversal occurs when one passes from planar curves to spatial curves concerning the handedness required for a C^2 splice. In the planar setting (spirals), the data to be matched at an intersection consists of position, unit tangent, and signed curvature, and because curvature is the only first-order invariant beyond the tangent, one free phase parameter suffices to adjust its magnitude but not its sign. Consequently, two offset spirals can be C^0-glued iff their curvature signs are opposite).

In the spatial setting (helices), a space curve adds a second scalar invariant (torsion), whose sign records right-handed versus left-handed screw motion. The gluing constraints now depend on the relative placement of the helical axes:

Proposition 1.4 (Rules for Smooth Gluing of Helices)**.** *Let:*

$$\gamma_i(t) \;=\; c_i + R_i\big(\cos t\,\mathbf{u}_i + \sin t\,\mathbf{v}_i\big) + p_i\,t\,\mathbf{e} \quad (i = 1, 2)$$

be two circular helices in $\mathbb{R}^3$ *whose axes are parallel to the same unit vector* $\mathbf{e}$*.*

Define curvature and torsion in the usual way:

$$\kappa_i = \frac{R_i}{R_i^2 + p_i^2}, \qquad \tau_i = \frac{p_i}{R_i^2 + p_i^2},$$

so $\operatorname{sign}(\tau_i)$ *records right- or left-handedness.*

(a) ***Coaxial case.*** *If the centres* c_1, c_2 *differ only along the common axis* $(c_2 - c_1) \parallel \mathbf{e}$*, then one can splice* γ_1 *to* γ_2 *by a* C^2 *curve through an intersection point* P iff

$$\tau_1 + \tau_2 = 0, \quad \textit{equivalently} \quad \operatorname{sign}(\tau_1) = -\operatorname{sign}(\tau_2).$$

(b) ***Offset-axis case.*** *If the axis directions coincide but the centres are displaced sideways,* $(c_2 - c_1) \perp \mathbf{e}$*, then a* C^2 *splice exists* iff

$$\tau_1 = \tau_2, \quad \textit{equivalently} \quad \operatorname{sign}(\tau_1) = \operatorname{sign}(\tau_2).$$

Proof sketch. At the chosen meeting point P we must match three items:

(a) Position P itself.

(b) Unit tangent directions.

(c) Signed curvature vector, i.e., curvature magnitude and the normal direction determined by torsion.

For coaxial helices, only one phase angle (say t_1) can be changed independently; this lets us match the curvature magnitude but forces the torsion signs to cancel. With offset axes, we have two independent phase angles (t_1, t_2), giving enough freedom to align the normal planes provided the torsion signs agree. A short trigonometric computation makes these statements precise. □

Remark 1.7. Passing from $\mathbb{R}^2$ to $\mathbb{R}^3$ promotes the matching problem from the 1-jet to the 2-jet bundle: curvature sign is now unsigned (since $\kappa > 0$), while torsion sign becomes the relevant orientation datum. Whether this new constraint binds or is absorbed by an extra phase degree of freedom hinges on the axial configuration. In effect, the added dimension introduces a framing (normal-binormal plane) whose sign can obstruct or permit the splice depending on how many phase parameters survive the geometric symmetry.

Proposition 1.5 ($THM_4 \cong V_4$)**.** *The group of transformations generated by the Chirality Swap (X) and Orientation Swap (Y) operators, which acts on the set of four canonical* THM *types, is isomorphic to the Klein four-group,* V_4*.*

Proof. Let $H_{\text{types}} = \{\text{Type 1}, \text{Type 2}, \text{Type 3}, \text{Type 4}\}$, representing the four configurations of unit helices determined by combinations of chirality (Same vs Opposite) and orientation (Same vs Opposite).

Define the following transformations acting on H_{types}:

- Identity I: Trivial transformation. $I(T_i) = T_i$ for all $T_i \in H_{\text{types}}$.
- Chirality Swap X: Swap the chirality relationship (Same $\leftrightarrow$ Opposite), while preserving orientation.
- Orientation Swap Y: Swap the orientation relationship (Same $\leftrightarrow$ Opposite), while preserving chirality.
- Composite $Z = X \circ Y$: Acts by applying both swaps.

The action on the four manifold types proceeds as follows:

Type	(Chirality, Orientation)	X	Y
H_1	(O, O)	H_3	H_2
H_2	(O, S)	H_4	H_1
H_3	(S, O)	H_1	H_4
H_4	(S, S)	H_2	H_3

Composition rules:

$$\begin{aligned} Z = X \circ Y = Y \circ X \\ Z(H_1) = X(Y(H_1)) = X(H_2) = H_4 \\ Z(H_2) = X(Y(H_2)) = X(H_1) = H_3 \\ Z(H_3) = X(Y(H_3)) = X(H_4) = H_2 \\ Z(H_4) = X(Y(H_4)) = X(H_3) = H_1 \end{aligned}$$

All non-identity elements have order 2:

$$X^2 = I, \quad Y^2 = I, \quad Z^2 = (X \circ Y)^2 = X^2 \circ Y^2 = I$$

Pairwise compositions:

$$X \circ Y = Z, \quad Y \circ X = Z \quad \Rightarrow \quad X \circ Y = Y \circ X$$

$$X \circ Z = X \circ (X \circ Y) = (X \circ X) \circ Y = I \circ Y = Y$$

$$Y \circ Z = Y \circ (X \circ Y) = (Y \circ Y) \circ X = I \circ X = X$$

The set of transformations $G = \{I, X, Y, Z\}$ forms a group under composition. All elements have order 1 or 2, and the group is abelian. Thus, $G \cong V_4$. □

Table 2 of the calculated FSG progressions via the GAP has been provided for ease of reference.

Table 2: Group Progression of the THM_4 Family

Group Structure	**Order**	**Defined by...**	**Type**
$V_4 \cong C_2 \times C_2$	4	Chirality and orientation combinations of the 4 canonical THM_4 types	abelian
$D_4 \cong (C_2 \times C_2) \rtimes C_2$	8	Chirality flips + swapping identity of the two helices	non-abelian
C_2^4	16	Independent flips of $(\chi_A, \chi_B, d_A, d_B)$ for each unit helix	abelian
$C_2^4 \rtimes C_2$	32	Above + helix identity swapping symmetry	non-abelian
$(C_2^4 \rtimes C_2) \times C_4$	128	Adds 90° global rotation symmetry around z-axis	non-abelian
$(C_2^4 \rtimes C_2) \times C_{4h}$ $\cong (C_2^4 \rtimes C_2) \times (C_4 \times C_2)$	256	Full group including xy-plane reflection symmetry (σ_h)	non-abelian

Table 3: Generators of the THM_{256} Symmetry Group

Generator	**Action**
c_1	Flip chirality of Helix A
c_2	Flip chirality of Helix B
d_1	Flip orientation of Helix A (up/down)
d_2	Flip orientation of Helix B (up/down)
S	Swap identity of Helix A and Helix B
R_{90}	Global 90° rotation of the twin helix around the z-axis
σ_h	Reflect the entire twin helix across the xy-plane

In addition, the generators of THM_{256} and lower orders are shown in Table 3. These symmetries exhibit a progression toward non-abelian behavior as more generators and state permutations are introduced. Preliminary computations suggest a rapidly growing group order (originally estimated at 256, but more accurately in the range of 1028 or higher when accounting for all independent orientation and chirality swaps between the two helices).

We note that this group structure may not be unique to the present construction. Similar generator sets appear in various symmetry and braid group contexts. The explicit relations and full structure of THM_4, including its precise order, are left open for further investigation, ideally by those with more expertise in computational geometric group theory.

We will now explore the topology of the THM. A single helical solid is a compact 3-manifold whose boundary, ∂H, is a smooth 2-manifold homeomorphic to a sphere. This boundary can be decomposed into the helical tube, the apical cap, and the base disk. The circular edge where the base disk meets the tube is a feature of this geometry. When we glue together the bases of two such unit helices in a proper manner, we form a larger compact 3-dimensional manifold with a single boundary. The original bases become an internal surface, and the boundary of the new THM consists of the original helical side surfaces and the tapered tips. THM is homeomorphic to D^3 in topological terms, giving genus 0.

Singularities at the apex are indeed a threat if the volume of the helix were to taper suddenly at the end to a point. But note that we have eliminated them earlier in the paper by way of a non-linear $\rho(\theta)$ volume decay and diffeomorphisms that produce smooth apex to THM's unit helices.

Definition 1.9 (Singularity–Smoothness Principle)**.** In the geometric–computational framework, every computational state is realized by a smooth manifold together with a prescribed flow. A necessary condition for logical consistency and well-posed evolution is the absence of geometric singularities. Concretely, the manifold and its flow must possess at least C^0 differentiability so that position and tangent fields vary continuously, guaranteeing the existence and uniqueness of solutions to the associated ordinary differential equations.

Definition 1.10 (Minimal Smoothness by Computational Regime)**.** The following differentiability thresholds are established:

- **Abelian Regimes** (Symbolic and Quasi-Fluidic) — For stable logic and deterministic integration, it suffices that each twin helix manifold be C^0; continuous tangents ensure well-defined local ODEs. Higher regularity (C^1 or smoother) is preferred but not strictly required.
- **Non-Abelian Regimes** (Fully Fluidic and Gauge) — Dynamics now depend explicitly on curvature and higher-order invariants. Hence, a minimum of C^2 smoothness is mandated, with higher regularity (C^k, $k \geq 3$, or analytic) desirable for robust geometric evolution and topological stability.

Remark 1.8. The Singularity–Smoothness Principle establishes that geometric smoothness is equivalent to logical consistency. The minimal smoothness requirements specify the differentiability needed in each computational regime, ensuring that every geometric computation proceeds without singular breakdown.

2 Geometrically Computable Manifolds—Achieving Turing Completeness in $\mathbb{R}^4$

We now begin the construction of a Turing Machine ("TM") from a lattice of THM. The feasibility of this approach rests on the fact that THM, unlike most manifolds, provides a rich but controlled symmetry space. It is precisely because most well-known shapes in this metric (e.g., circles, tori, spheres) possess too much symmetry—not too little—that they fail to offer the structural asymmetry needed for computational behavior. The THM construction breaks this impasse.

2.1 Geometrically Computable Manifolds

Definition 2.1 (Geometrically Computable Manifold)**.** A **GCM** on a smooth ambient space is:

$$\mathcal{G} \;=\; \big(\,\mathcal{M},\; \{R_i\}_{i\in I},\; \pi : E \to \mathcal{M},\; \mathcal{I},\; \mathsf{Ev}\,\big)$$

consisting of:

(a) **Smooth Manifold** $\mathcal{M}$ — a smooth manifold (often a finite working region, possibly with boundary). A finite family of compact submanifolds $\{R_i\}_{i\in I}$ (cells) covers the working region and provides locality (finite overlap pattern).

(b) **State bundle** $\pi : E \to \mathcal{M}$ — a smooth fiber bundle with finite dimensional fibers. A *global state* is a section $\sigma \in \Gamma(E)$. Discrete/symbolic, toroidal/phase, and integer-valued components may all live in the fiber.

(c) **Interfaces** $\mathcal{I}$ — for each nonempty overlap $R_i \cap R_j$, an interface map:

$$I_{ij} :\; \big(\sigma|_{R_i},\, \sigma|_{R_j},\, \text{local data}\big) \longmapsto \big(\sigma'|_{R_i},\, \sigma'|_{R_j}\big),$$

attached to a portion of $\partial R_i \cap \partial R_j$. Interfaces couple neighboring cells and enforce locality: updates at R_i depend only on σ over R_i and finitely many overlaps $R_i \cap R_j$.

(d) **Evolution** Ev — either:

- discrete time: an update operator $U : \Gamma(E) \to \Gamma(E)$ that is local with respect to $\{R_i\}$, or
- continuous time: a flow of sections $(\sigma_t)_{t\in\mathbb{R}}$ solving $\dot\sigma_t = F(\sigma_t)$ for a smooth, fiber-wise vector field F over $\mathcal{M}$;

possibly accompanied by a base diffeomorphism $\Phi_t : \mathcal{M} \to \mathcal{M}$. In either case, evolution is nonsingular, it exists for all steps/times on admissible initial states, and remains in a fixed finite energy subset of E.

So what is a GCM, really? A GCM is a manifold that can encode its computable properties. A single THM can qualify as a GCM. A lattice of THM is also a GCM—just with more computing capacity. We call $\mathcal{G}$ a GCM when its evolution is local, interface-driven, and closed under composition: finite products of such systems and gluing along matched interfaces yield new systems of the same kind.

2.2 Geobits—Bits, Geometric Bits, and State Space

Definition 2.2 (Geometric Bits — Symbolic and Enriched)**.** Let $\mathbf{S}_{\mathrm{GCM}}$ be the category of state space embedding all GCMs will use. Fix an ambient manifold region $\mathcal{M} \subset \mathbb{R}^n$ and a twin–helix template H (a pair of embedded tubular helices) together with a framing: a preferred axial direction $\hat z$ and a reference scale r_0. Let $\mathrm{Emb}^{\mathrm{fr}}(H, \mathcal{M})$ denote the set of framed embeddings $e : H \hookrightarrow \mathcal{M}$ modulo framed ambient isotopy that preserves the axial framing and the scale r_0 so that radius and orientation remain part of the state.

Let $H_{\mathrm{ext}} \le O(n)$ be a chosen ambient symmetry group (allowing reflections so that chirality changes are representable) and let H_{int} be a finite internal symmetry group acting on the labels of the twin (e.g. the swap of the two helices). Set $G := H_{\mathrm{ext}} \times H_{\mathrm{int}}$, acting on $\mathrm{Emb}^{\mathrm{fr}}(H, \mathcal{M})$ by pre/post–composition.

(a) **Symbolic geobit** — Define the symbolic geobit state set:

$$\mathbf{S}_{\mathrm{sym}} \;:=\; \Big\{\, [e; \chi, \omega] \;\Big|\; e \in \mathrm{Emb}^{\mathrm{fr}}(H, \mathcal{M}),\; \chi,\; \omega \Big\} \Big/ G,$$

where $[e; \chi, \omega]$ denotes a framed embedding decorated by chirality (χ) and axial orientation (ω) relative to $\hat{z}$, and the quotient is by the G–action, so states are defined up to global symmetry. A *symbolic geobit* is an element $s \in \mathbf{S}_{\text{sym}}$. The finite subgroup generated by "flip chirality" C, "flip orientation" D, and (optionally) "swap" $S \in H_{\text{int}}$ acts on $\mathbf{S}_{\text{sym}}$; in the canonical twin–helix unit this induces a $V_4 \cong \mathbb{Z}_2 \times \mathbb{Z}_2$ action on the labels (χ, ω).

(b) **Enriched (quasi–fluidic) geobit** — To allow compact phase and quantized charges, extend the label by a finite–type fiber:

$$P \;\cong\; (S^1)^m \times \mathbb{Z}^p \times F,$$

where the S^1–coordinates encode bounded phases (e.g. axial angle), the $\mathbb{Z}$–coordinates encode twist/winding/linking numbers, and F is a finite set of discrete tags (e.g. mode, parity). Set

$$\mathbf{S}_{\text{enr}} \;:=\; \Big\{ [e; \chi, \omega, p] \;\Big|\; e \in \text{Emb}^{\text{fr}}(H, \mathcal{M}),\ \chi,\ \omega,\ p \in P \Big\} \Big/ G,$$

with G acting equivariantly on the S^1–factors (by rotation) and preserving the $\mathbb{Z}$–charges. An *enriched geobit* is an element of $\mathbf{S}_{\text{enr}}$. The G–action need not be transitive; its orbits in $\mathbf{S}_{\text{enr}}$ are the *geobit types*. The symbolic case is recovered by taking $m = p = 0$ and F trivial.

In both cases, the geobit is a G–set (not necessarily a group): the finite operations generated by C, D (and S when present) act on states, yielding the canonical V_4 action in the symbolic unit. The enrichment by P provides the quasi–fluidic degrees of freedom (compact phases and integer counters) used elsewhere in the text.

For the time, we work only with S_{sym}. Now, let: [2]

$$(H_1, H_2, H_3, H_4) \mapsto (00, 01, 10, 11)$$

where each H_n is a configuration from the THM_4 symmetry schema defined in Table 1, preserving the order of the original type chart.

We assign bit values based on symmetry: if the chirality and orientation of the unit helices are the same, assign bit value 1; if they differ, assign 0. This yields a total of 2 bits per THM_4 configuration.

Definition 2.3 (Geobit Capacity)**.** Let THM_O be a GCM whose internal structure is governed by an abelian group O, acting purely on intrinsic symmetries (i.e., excluding extrinsic spatial transformations such as rotations or reflections). Then:

$$\text{Geobit capacity} = \log_2 |O|$$

That is, the order of the symmetry group O determines the number of encodable bits in the resulting GCM under classical logic.

Example 2.1 (Embedding Relationships for Helices). Let G be a finite group acting on a GCM configuration space, and let $O \subseteq G$ be the subgroup corresponding to encodable internal transformations (e.g., chirality, orientation). Define the number of distinguishable states by:

$$\text{Geobit capacity} := \log_2 |O|.$$

For example:

(a) In THM_{16} (governed by C_2^4), each helix can encode 4 bits total: 2 bits from chirality and 2 from orientation.

(b) In THM_4, this reduces to 2 bits, since we encode only global relationships (i.e., quotient by identity swapping and pairing symmetries).

(c) In THM_2, assuming only orientation is swappable, we drop to 1 bit per configuration.

Thus, a GCM based on THM_O can operate in binary, quaternary, or hexadecimal—depending on the order of its encoding group O.

Table 4 provides a reference encoding the relationships.

[2] Magic happens here.

This leads us to a key structural pattern in the group progression. Let G act on configuration space X, then:

Abelian case: If $G \cong (C_2)^n$, then $|X| = 2^n$ and each orbit is distinct.
Non-abelian case: If G is non-abelian, then $|X/G| < |X|$; the quotient space is smaller due to collisions under conjugacy.

Remark 2.1. Why the focus on abelian vs non-abelian groups?

In an abelian group, every state can be constructed as the direct sum of independent generators. That is, every geobit state is uniquely addressable by an n-tuple over $\mathbb{C}_2^n$. There are no collisions. The total number of distinguishable geobit states is exactly 2^n, and the encoding is non-redundant.

In contrast, a non-abelian group acts less cleanly. Its action "mixes" states: some group elements act non-trivially, creating identifications among configurations. Orbits form under conjugation or subgroup stabilizers, and the full state space becomes quotient-like. Redundancy becomes intrinsic—some geobit states are indistinguishable under the group action. As a result, the number of distinct encodable states in the orbit space may be strictly less than the order of the group itself.

This distinction becomes crucial when interpreting a GCM as a computational substrate: abelian structure maximizes symbolic resolution, while non-abelian structure imposes constraints that must be managed or exploited.

Table 4: Relationship Between Geobits and Their Groups

Group	**Order**	**Dimension**	**# Bits**	**Comments**
C_2	2	2	1	Classical bit
$(C_2)^2 = V_4$	4	4	2	Base level 2-geobit
$(C_2)^4$	16	4	4	4-geobit - Embeds in SO(5)
$(C_2)^8$	256	8	8	8-geobit - Embeds in SO(9)
D_4	8	3	≤ 3	non-abelian, simple global swap
$D_4 \ltimes (C_2)^4$	128	≥ 5	< 8	non-abelian, global rotations

Remark 2.2. The embedding dimension for a THM symmetry group is the minimal integer n such that all distinct geometric configurations corresponding to the elements of the group can be realized as non-self-intersecting embedded submanifolds in Euclidean space.

Given that there are 2^4 unique internal configurations of each THM in the metric space $\mathbb{R}^4$, we could, if desired, define each individual state as a separate base unit—using standard hexadecimal notation (e.g., `0x0` through `0xF`). Although the notion of a hexadecimal-based computer is intriguing, we will mostly focus on THM_4, which runs naturally in quaternary due to its $\mathbb{Z}_2 \times \mathbb{Z}_2$ structure.

It is important to note that global reflections or full rotational symmetry groups do not count toward the intrinsic geobit capacity in $\mathbb{R}^4$. These operations require action on the global geometry and do not alter the intrinsic state of the unit helices. Thus, group orders beyond 16 that arise from external symmetry do not increase state space in a computable sense.

Had we remained in $\mathbb{R}^3$, the maximum distinct internal configuration would be just 2^1, governed by the group C_2. Therefore, the dimensionality of the ambient metric space directly impacts the maximum geobit capacity.

Definition 2.4 (Notation for Specific Geobit Information Capacity)**.** Let $\mathcal{G}$ be a GCM associated with a symmetry group O. Then the classical information capacity (in bits) of a geobit in $\mathcal{G}$ is defined as:

$$I(O) := \log_2 |O|,$$

where $|O|$ is the order of the group of allowable geometric transformations within $\mathcal{G}$.

We refer to this as an "n-geobit," where $n = I(O)$ gives the classical bit-equivalent capacity. This naming system enables comparisons across different GCMs, as well as with classical and quantum bits.

For example:

$$I(C_2^5) = 5 > I(V_4) = 2.$$

Such comparisons allow us to define geobit inequalities or equivalences in information-theoretic terms, purely from the group structure. This notation may be useful when working with layered geometries or hybrid encoding schemes across different GCM classes.

Table 5: Quantum vs Geometric State Space

Quantum Qubit	**Geometric Geobit**
Bloch sphere $SU(2)$	Polyhedra/Polytopes formed from $SO(n)$
Superposition	Hidden Variable State System
Rotation gates (Pauli-X, Y, Z)	Flips, rotations, and deformations of substructures
Born Rule application collapses state	Reading chirality/orientation collapses state gradually
1 classical bit of storage	Group order/metric space determines classical bits of storage

The table 5 succinctly compares geobits with qubits via conceptual relations. [3]

Regarding the construction of a "Hidden Variable State System." This term demands some explanation. Recall that the superposition of a qubit is written as a state vector:

$$\alpha|0\rangle + \beta|1\rangle, |\alpha|^2 + |\beta|^2 = 1$$

In contrast, a geobit has structured ambiguity via chirality and orientation coupling, as in our case of the twin helix topology. It has no amplitudes, so it is deterministic but not locally resolved until a fixed point is chosen. Knowing information about one side of the geometric substructure helps an observer narrow down the possible overall state of the entire system. So until one "reads" the manifold, the configuration could be thought of as simultaneously "leaning toward" multiple classical states. This resembles what is known in physics as a *hidden variable system*.

Definition 2.5 (Hidden Variable State System and Notation)**.** A hidden variable state system is a state system in which the values of individual bits or local states may be inaccessible or indeterminate from an external perspective, yet their configuration exerts structured, deterministic influence on the global system. Information about the system increases as more local structure is revealed, with local geometry shaping the overall state. From the outside, it may appear probabilistic, but the determinism is fully set in the geometric configuration so as not to violate Bell's Inequalities. Superposition-like ambiguity arises not from true indeterminacy, but from our inability to fully resolve or observe the internal geometry without disturbing it.

Definition 2.6 (Notation for Hidden Variable State System)**.** We define a *Hidden Variable Ket* ("HV-ket") to be a geometric computational state descriptor of the form

$$|\alpha|\beta\rangle_G$$

Where:

(a) α is the classically accessible bit configuration, potentially masked.

(b) β is the hidden morphism or transition operator responsible for the current or next state.

(c) G the governing group or groupoid of allowable morphisms.

I have chosen this $|\alpha|\beta\rangle$ ket-style notation in the spirit of following the tradition of qubits, but with internal delimiters in the ket to show that, unlike in qubits, in this case α and β are not two separate vectors.

Example 2.2 (Practical Use of HV-ket). Say we have H_{AB}. We are starting out operating in C_2^4. We know that the chirality/orientation of H_A is Right/Up. Arbitrarily, we define Right/Left-handedness = 1/0 and Up/Down orientation = 1/0. The bit structure is sequential with respect to the unit helices.

$$|C_A, O_A, C_B, O_B|\beta\rangle_{C_2^4}$$

[3] Recall Holevo's Theorem.

To avoid confusion in α, we explicitly include masked bit unknowns as X in the visible classical state. One can imagine a scenario where we knew the orientation of both helices as Up but not their chirality. It would be confusing to write:

$$|11|\beta\rangle_{C_2^4}$$

Without knowing what bits were what. So we can then write the state of H_{AB} as:

$$|11XX|\beta\rangle_{C_2^4}$$

Let's say we decided to measure H_B, and found out it was Left and Up. Then we could write out the updated 4-geobit as:

$$|1101|\beta\rangle_{C_2^4}$$

Since we know the full state, the X's disappear. We define the projection map $\pi : C_2^4 \to V_4$, where:

$$\pi(C_A, O_A, C_B, O_B) = (C_A \oplus C_B, O_A \oplus O_B)$$

So under V_4, this would be quotiented down to:

$$|10|\beta\rangle_{V_4}$$

However, this is only possible since we know the state of all the bits. If we did not know the geometry states of H_B, then the masks are inherited as such:

$$|XX|\beta\rangle_{V_4}$$

or if, say, the bit structure was different, it might appear as:

$$|1X|\beta\rangle_{V_4}$$

The V_4 group traditionally uses the notation (a, b, ab, e) for its generators. Let's say b flips the orientation. We can write the β delimited part of the ket as a group action that is "primed" to cause a state change in α on H_A.

$$|11|b_A\rangle_{V_4}$$

And then we can also use arrows to display group actions on specific HV-kets.

$$|11|b_A\rangle_{V_4} \xrightarrow{b} |10|e\rangle_{V_4}$$

The identity generator being the "default" state of β after the group action.

2.3 The GCM Lattice

Definition 2.7 (GCM lattice)**.** A *GCM lattice* is a finite or locally–finite assembly of GCMs arranged over a graph. Formally, it is the data:

$$GL \;=\; \big(\,\mathcal{L},\; G=(I,E),\; \{\mathcal{G}_i\}_{i\in I},\; \{\iota_i\},\; \mathcal{I}^{\mathrm{lat}},\; \mathsf{Ev},\; \mu\,\big)$$

with:

(a) **Ambient region $\mathcal{L}$** — smooth manifold (often a bounded working region) into which cells will be embedded.

(b) **Lattice graph $G = (I, E)$.** A finite or locally–finite graph (e.g. a box of $\mathbb{Z}^d$) with uniformly bounded degree; vertices index the cells, edges encode adjacency.

(c) **Cells $\{\mathcal{G}_i\}$** — For each $i \in I$, a GCM $\mathcal{G}_i = (\mathcal{L}_i, \{R^i_\alpha\}, \pi_i : E_i \to \mathcal{L}_i, \mathcal{I}_i, \mathsf{Ev}_i)$ as in Def. 2.1.

(d) **Placements $\{\iota_i\}$** — Smooth embeddings $\iota_i : \mathcal{L}_i \hookrightarrow \mathcal{L}$ with pairwise disjoint interiors. The support of cell i is $\mathrm{supp}(i) := \iota_i(\mathcal{L}_i)$.

(e) **Lattice interfaces $\mathcal{I}^{\mathrm{lat}}$** — For each edge $(i, j) \in E$, a family of interface maps attached to portions of $\partial\mathrm{supp}(i) \cap \partial\mathrm{supp}(j)$,

$$I_{ij} :\; \big(\sigma|_{\mathrm{supp}(i)},\, \sigma|_{\mathrm{supp}(j)},\, \text{local flow data}\big) \;\longmapsto\; \big(\sigma'|_{\mathrm{supp}(i)},\, \sigma'|_{\mathrm{supp}(j)}\big),$$

which are local (depend only on a finite stencil around i, j) and equivariant with respect to the chosen cell symmetries.

(f) **Evolution** Ev — Either (i) a discrete, synchronous update $U : \Gamma(E) \to \Gamma(E)$ on the global state bundle $E := \bigsqcup_{i\in I} \iota_i E_i$, or (ii) a continuous flow of sections $(\sigma_t)_{t\in\mathbb{R}}$ solving $\dot{\sigma}_t = F(\sigma_t)$ and compatible with all interfaces. Evolution is local and non-singular on admissible initial states.

(g) **Symbolic observation** μ — A readout map $\mu : \Gamma(E) \to \Sigma^N$ that, at each cell i, selects a coding $q_i : E_i \to \Sigma^{n_i}$ of its geobits; the total capacity is $N = \sum_{i\in I} n_i$.

We say a *GL* is *computationally coherent* if the symbolic dynamics induced by the geometric evolution commutes with readout, i.e., there exists $\Phi_s : \Sigma^N \to \Sigma^N$ with:

$$\mu\big(U(\sigma)\big) \;=\; \Phi_s\big(\mu(\sigma)\big) \quad \text{(discrete time)}, \qquad \text{or} \qquad \frac{d}{dt}\,\mu(\sigma_t) \;=\; F_s\big(\mu(\sigma_t)\big) \quad \text{(continuous time)}.$$

Remark 2.3. The architecture of geometric computability is decidedly a non–Von Neumann one. A GL is necessary to compute programs that are not trivial, since one can't discretely read out results on a single GCM with symbolic or quasi-fluidic computation, not without a projection layer or a symbolic "slice." In a GCM there is no separation of program and data, no instruction pointer, or fetch–decode–execute cycle, the geometry is the computation. Computation is intrinsic, continuous, and spatially distributed. While one can clearly see the similarity to cellular automata in the discrete case, parallels can be drawn to neural fields or reaction-diffusion systems if one requires an outside example.

Example 2.3 (Twin Helical Chain as Degenerate GCM Construction). Consider the following construction for encoding an n-bit geometric computer within a single manifold. Start with a 2-manifold homeomorphic to a solid cylinder, with one end smoothly rounded ("bullet-shaped") and the other a flat disk boundary. Place this horizontally along axis h_1.

Deform this tube into a helical shape about a vertical axis v_1. After a prescribed number of turns, direct the tube laterally, then wrap it down along a second vertical axis v_2, returning to h_1. This creates a Type 4 THM with smooth joins at what were previously the helix apexes.

At the end of h_1, adjoin a flat-disk boundary. You may now glue (via diffeomorphism) as many such helical segments as desired along a single axis, each encoding a bit (geobit). Cap the chain with a final bullet-shaped end.

The entire construction is a single, smooth manifold, a "Twin Helical Chain" encoding arbitrary geobit capacity. However, because the closed-loop twin helices must twist and untwist in tandem, their chiralities and orientations are not independently manipulable, so bit transitions are collective, not local.

Remark 2.4. **Warning.** The Twin Helical Chain is a single manifold GCM that is technically a *degenerate case* since it achieves arbitrary geobit capacity though purely symbolic quantization only in a single manifold, causing the lattice (which by definition recall is made a series of *discrete* GCMs) to become 1×1. Note that there is technically a second degenerate case of a GL consisting of a single GCM in a 1×1 matrix that still achieves infinite-geobit capacity. See discussion of the "classical" measurement problem in section 4.13. Likewise, this is simply not what we observe in nature, see 5.6.

2.4 FSM Encoding for Twin Helices

Finite state machines (FSMs) are the building blocks of classical computation. An FSM consists of:

(a) $\mathbf{S}$ — A finite set of states.

(b) $\mathbf{\Sigma}$ — An input alphabet.

(c) $\delta : S \times \Sigma \to S$ — A transition function.

(d) s_0 — An initial state.

(e) F — A set of final (accepting) states or outputs.

Proposition 2.1 (FSM Geometric Embedding)**.** *In the symbolic mode, every GCM admits an interpretation as a self-contained FSM fragment, with its internal and boundary structure encoding computational components.*

Proof. The classical FSM tuple $(S, \Sigma, \delta, s_0, F)$ may be represented geometrically as follows:

(a) $S \cong X \times \Omega$ — Each unit helix has both a chirality and an orientation. The product $X \times \Omega$ forms the finite configuration space of states. (If the GCM structure is larger, the state space grows as a finite product accordingly.)

(b) Σ — Inputs correspond to discrete, physically realizable perturbations or boundary signals. These are not abstract symbols, but geometric interactions such as boundary twists, deformations, or coupling signals. Allowed input types can be classified into a finite set, providing a physical basis for Σ.

(c) δ — Transitions are governed by deterministic local dynamics under the construction rules of the GCM. This includes interactions such as twist propagation, chirality flip, or boundary constraint. These local rules induce a transition function δ that can be compiled into a classical FSM transition table, except here, every transition is grounded in a geometric move.

(d) s_0 — The initial state corresponds to a fully masked or idle geobit—one with unknown (or fixed) chirality and orientation. This is represented by an idle configuration, using quantum-inspired notation, an HV-ket such as $|\alpha|\beta\rangle_G$, where α is the chirality and orientation configuration and β the set of allowable group morphisms.

(e) F — The output is the final geometry of the GCM (or, in the lattice case, the geometry of neighboring GCMs). While FSMs do not always have a unique output, in a composable computational system, the final geometric configuration of the GCM serves as the computation's result.

□

Table 6 compares classic computational constructions with symbolic regime geometrical logic.

Table 6: Classic vs Geometric Symbolic Computation

Feature	**Classic Computability**	**Geometric Computability - Symbolic Mode**
Objects	Strings, tapes, numbers	Smooth manifolds, embedded surfaces
Operations	Symbol rewriting, Turing steps	Chirality and orientation flips
State	Sequence of symbols	Geometric configuration states of substructures
Constraints	Logical consistency	Topological smoothness
Groups acting	Discrete automata	Discrete subgroups of diffeomorphisms

Theorem 2.2. *A lattice of GCMs can achieve Turing completeness with symbolic communication.*

Proof. Recall that:

$$THM_4 \cong V_4 \cong C_2 \times C_2$$

The universality of Rule 110—proven by Cook [3] and originally conjectured by Wolfram [4]—establishes that a cellular automaton based on a cyclic tag system is Turing complete. Since THM_4 naturally carries two cyclic groups, its state space is sufficiently rich for this construction.

Define the state space for each THM as a 2-geobit, encoding 2 classical bits of information. Let L be a GL. Each cell in L contains a THM_4 manifold (with unit helices A and B), which communicates symbolically with its adjacent cells. Suppose there is a readout layer μ, with one pixel per THM. Each pixel μ_i displays as black or white based on chirality agreement:

$$\text{Readout}(\mu_i) = \begin{cases} \blacksquare & \text{if Chiral}(A) = \text{Chiral}(B) \\ \square & \text{otherwise} \end{cases}$$

Assume discrete time $t \in \mathbb{N}$. At each time step:

(a) Each cell H_i evaluates the orientation states of its neighbors H_{i-1}, H_i, H_{i+1}.

(b) Applies the local update rule to determine the new bit b_i^{t+1}.

(c) Updates its Helix A orientation and/or chirality accordingly.

Transition rules for Rule 110 use the group generators as defined in Section 1.4.

Bit encoding:

$$b_i^t \in \{0,1\}$$
$$\text{Orient}_i^t = (\text{Orient}_A^t, \text{Orient}_B^t)$$
$$\text{Chiral}_i^t = (\text{Chiral}_A^t, \text{Chiral}_B^t)$$

where $b_i^t = 1$ if $\text{Orient}(A) = \text{Orient}(B)$, and $b_i^t = 0$ otherwise.
We assume Helix B is fixed, and only Helix A may change.

Rule 110 update law:

$$\delta(b_{i-1}^t, b_i^t, b_{i+1}^t) = \begin{cases} 1 & \text{if } (b_{i-1}^t, b_i^t, b_{i+1}^t) \in \{110, 101, 011, 010, 001\} \\ 0 & \text{otherwise} \end{cases}$$

Group actions update law:

$$b_i^{t+1} = \delta(b_{i-1}^t, b_i^t, b_{i+1}^t)$$
$$\text{If } b_i^{t+1} \neq b_i^t: \quad d_1$$
$$\text{If } b_i^{t+1} = 0: \quad c_1$$

Here, d_1 and c_1 are the group generators acting on Helix A.

Given that L is of sufficient size to simulate both "boats" and "gliders" (the minimal universal structures in Rule 110), it follows that a lattice of THM_4 manifolds is Turing complete. □

2.5 Visualizing the Shape of Computation

A regular bit naturally lives on a Boolean number line, existing as 0 or 1. A qubit, as a state vector, naturally lives on the Bloch Sphere. Where does a geobit live? The answer is tentative. Like we established, the group operations determine the number of bits a quantized GCM uses.

In a GCM, one doesn't need complex amplitudes or wavefunctions, as the symmetry group of all possible states resembles a "discretized" Bloch Sphere. We will now turn to a geometric source to create a state space visualization of the geobits of THM_4. Polytopes can naturally be used to represent the state space of geobits. In this case, the vertices of the polytope are the THM states, and the edges are the group transformations.

Starting with V_4, D_4, they can fit easily into T, O. Although they encode the same number of bits, O represents an additional flip bit and requires additional edges. Now, however, we require a subgroup for $(C_2)^4$. Consider the binary octahedral group $2O$. This group, along with its brothers $2T$ and $2I$, is a double cover subgroup in the group of 3D point rotations SO(3). Given that we are primarily interested in representing a computational group of 16, a group of order matching 48 is the best choice for clean subgroup embedding via Lagrange's Theorem.

Or, so it would seem. But a not often mentioned theorem causes a paradoxical result that prevents clean embedding of the THM computational group model in a state space congruent to the dimensions that its geometry naturally lives in.

Proposition 2.3 (Minimal Embedding Dimension for THM_{16} Symmetry Group)**.** *A faithful embedding of the group THM_{16} (with symmetry group $(C_2)^4$) as a symmetry subgroup requires at least $SO(5)$.*

Proof. The Sylow 2-subgroups of the binary octahedral group $2O$ (a double cover of the octahedral rotation group in $SO(3)$) are isomorphic to the generalized quaternion group Q_{16} (the binary dihedral group $2D_4$). Q_{16} is non-abelian and contains elements of order 8, while $(C_2)^4$ is abelian and all non-identity elements have order 2. Moreover, $2O$ only has one central element of order 2 (corresponding to $-I$ in $SU(2)$), whereas $(C_2)^4$ requires 15 distinct elements of order 2. Therefore, $(C_2)^4$ cannot embed into $2O$ as a subgroup.

One might expect $SO(4)$ to be sufficient, since the geometry of THM_4 naturally lives in 4D. However, a direct analysis of the group's structure reveals this to be impossible. It is a standard result in the theory of Lie groups [5] that the largest possible elementary abelian 2-subgroup of SO(4) has rank 2, being isomorphic to the Klein four-group.

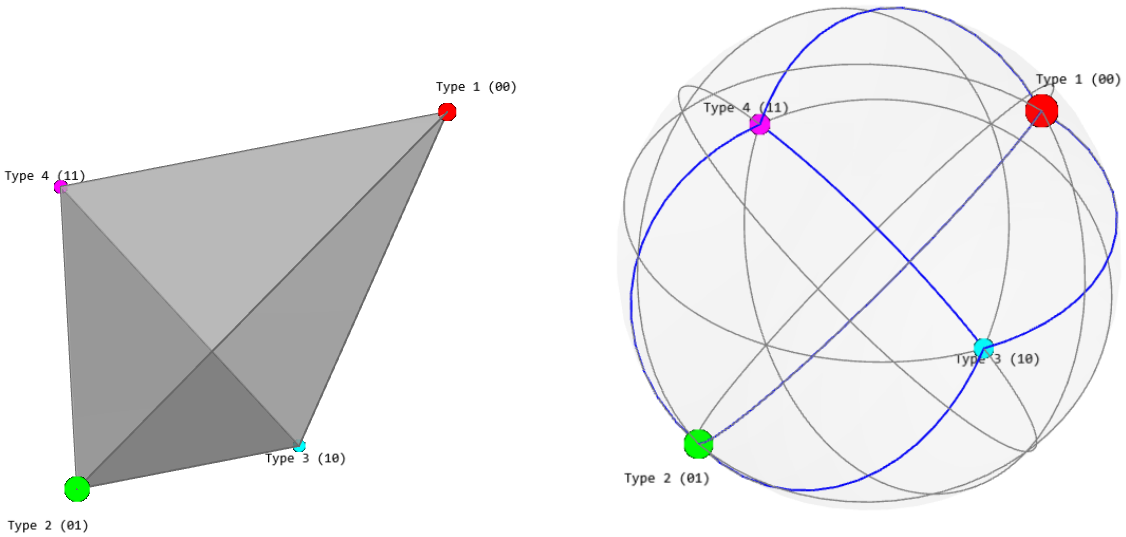
Hence, embedding $(C_2)^4$ as a symmetry group requires $SO(5)$ at minimum. Embedding $(C_2)^5$ would require $SO(6)$, but such a group does not arise in the THM progression, which becomes non-abelian above order 16 due to the use of semi-direct products.

Therefore, the state space embedding of THM_{16} necessitates a "dimensional jump"—manifesting emergent 5D algebraic behavior, despite the geometric construction living in 4D. [4] □

To embed all three original computational groups, we consider the full symmetry group of the 5-dimensional hypercube (the penteract), which is the Coxeter group B_5. This group has order $2^5 \cdot 5! = 32 \cdot 120 = 3840$. The relevant rotation subgroup, called the *demipenteract*, is $D_5 \cong (C_2^4) \rtimes S_5$, of order 1920. This group contains the diagonal $(C_2)^4$ symmetry and subgroups isomorphic to D_4 and V_4. The demipenteract has 16 vertices and provides a rigorous setting for realizing our algebraic structures in the context of an actual Lie group.

Since it is not possible to faithfully visualize a 5-dimensional shape, we settle for visualizations of V_4 and the regular tetrahedral group T (of order 12, embedded in $SO(3)$). In these visualizations, the polyhedral points correspond to the allowable states of the GCM, serving as the vertices of a 3D Cayley graph. The edges of the polyhedron represent transitions between these states. Figure 2a displays such a rendering, with colorings chosen to highlight these symmetries.

Remark 2.5. This "discretized" Bloch sphere can be constructed by representing the state space T as a spherical tessellation. In this approach, each vertex corresponds to one of the four fundamental geobit states, while edges represent single group-symmetry operations, and faces correspond to minimal closed loops of these actions. Notably, only four colors are required, one for each geobit state, making the discrete structure explicit. This construction provides a visual and combinatorial analogue of the quantum Bloch sphere, tailored to the discrete symmetry of the geometric bit model. Figure 2b illustrates this method. This discrete tessellation highlights the analogue between the continuous state space of quantum systems and the finite, symmetry-driven state space in geometric computation.



(a) V_4 Bit Group as a 3D Cayley Graph

(b) A "Discretized" Bloch Sphere - V_4 in (2 3 2)-sphere Form

Figure 2: Different Representations of the V_4 Group.

This spherical tessellation is composed of congruent Schwarz triangles; the specific numbering here is (3 2 3). All which are spherical triangles characterized by angles of $(\frac{\pi}{3}, \frac{\pi}{2}, \frac{\pi}{3})$. The vertices of these triangles on S^2 correspond to the points where axes of 3-fold, 2-fold, and another 3-fold rotational symmetry of the tetrahedron intersect the sphere.

Remark 2.6. The tetrahedral structure of the V_4 GCM state space is highly notable in light of results by Baez [7], who observed similar collapses in the quantum geometry of 3D vs 4D tetrahedra. However, in our case, this structure emerges entirely from combinatorial flow logic, with no reference to spin network

[4] See Feynman's *Lectures on Computation* [6, p. 111].

quantization. This suggests a deeper relationship between geometric expressiveness, computational state space constraints, and the topology of state polyhedra—one that may precede quantization itself.

2.6 The Logic of Geometry

GCMs can encode not only classical binary logic, but also three-valued logic and fuzzy logic. This is accomplished by introducing a third value to represent uncertainty, which is implemented by modifying the join angle between unit helices. This geometric logic can take two main forms:

(a) **Ternary / n-ary logic** — where bits take values in $\{0, U, 1\}$ (with U an "undecided" value not computed, but defined on $[0, 1]$), or more generally in $\{0, 1, \ldots, n-1\}$ for n-ary logic. For example, pentary logic uses values 0 through 4.

(b) **Fuzzy logic** — where bits may take values in $\{0, P, 1\}$, where P is a real number in $[0, 1]$, computed by a function such as $f(\theta) = \cos^2(\theta)$. The specific choice of function f determines the logic's behavior and can be tailored to computational requirements.

If the tangent surface between the two unit helices at the glue joint remains non-singular, then helices may be joined at arbitrary angles—not just 0° or 180°. This flexibility enables both n-ary and fuzzy logic encodings. For instance, simple three-valued logic is realized by a "trinary join" determined by a function:

$$f : (n \cdot 120^\circ) \mapsto \{0, U, 1\}$$

where n indexes the n-ary digit system, running from 0 to $n-1$. All angle measurements are taken from the positive z-axis, proceeding clockwise.

So far, we have deliberately excluded infinite groups from our analysis, for two distinct reasons:

(a) Allowing arbitrary join angles breaks the natural algebraic group structure of THM_4. Closure under group operations is lost when joins are not restricted to a finite set.

(b) The cross section of unit helices would be forced into a single conic (the circle), limiting geometric expressiveness and precluding the use of ovals or more general curves.

Both forms of logic, however, induce separate geometric, algebraic, and computational trade-offs. Three- or four-valued logic preserves finite group structure, since 120° or 90° rotations allow for finite algebraic closure. Fuzzy logic, by contrast, requires infinitesimal join rotations and destroys all vestiges of finite group structure, in addition to enforcing geometric isotropy.

Remark 2.7. Given that there exists a fine-grained tradeoff between the capacity for geometric expressiveness, algebraic group closure, and logical computational power, **this suggests the existence of a conservation law**. To draw a lightweight analogy—this is similar to Noether's Theorem of Conservation but for analog forms of computational logic. It should be noted that instead of dealing with the process of how symmetries give conserved quantities, it's more akin to a conservation law for symmetry *itself*. That is, it measures how one gives up potential possibilities in geometries to gain symmetry. I do not equate strict equivalence, as doing so with Notherian analogies without the presence of the correct algebraic structures (i.e. Lagrangian physics) is inherently dangerous. See Baez [8].

Lemma 2.4 (Geometric Logic Conservation Lemma)**.** *For any GCM G, there exists a conservation law between its algebraic and geometric computational capacities. Specifically, there is a constant K such that*

$$\mathcal{C}(G) + \mathcal{E}(G) \leq K$$

where $\mathcal{C}(\mathcal{G})$ denotes the algebraic closure of the logic gate set induced by $\mathcal{G}$ (i.e., which logical compositions remain valid under repeated joins or symmetries), and $\mathcal{E}(G)$ denotes a measure of geometric expressiveness (e.g., the number of distinct cross-sectional geometries supported by G). The constant K is determined by the topological and group-theoretic invariants of the system.

Here $\mathcal{C}(G)$ counts the $K(G)$-invariant gate clone and $\mathcal{E}(G)$ counts $K(G)$-orbit-distinguishable geometries at fixed readout; K depends only on topological and group-theoretic invariants of G. One path becomes a clean, algebraic infinite-state computation model. The other path devolves into an increasingly purely geometric, analog, continuous processor. A GL made of THM_4 sits at the cusp of this tradeoff. Its geometric expressiveness is quite high—as we will see. In classical physics, symmetry preserves energy and momentum, but Lemma 2.4 in GCT preserves computability and algebraic closure within a tightly constrained topological economy.

Remark 2.8. The essential point is that the symmetry of the system's configuration space directly induces an invariant quantity in the system's dynamics. Every symmetry break in topology introduces a logical ambiguity, and vice versa. Although it is difficult to visualize, given that we have only derived a single form of GCM, one can imagine higher-order manifolds where you can deform the joint outright into asymmetric 2-manifolds. These higher-order manifolds would devolve into pure fluidic computability, leaving discrete logic altogether and entering increasingly abstract and chaotic forms of analog computation. See 4.1 for direct visual explanation.

Definition 2.8 (Canonically Oriented Submanifold)**.** Let M be a smooth n-dimensional manifold and $\Sigma \subset M$ a smooth, connected k-dimensional submanifold. We say that Σ is canonically oriented if:

(a) **Orientable**— The tangent bundle $T(\Sigma)$ is orientable in the usual sense (there exists a nowhere-vanishing k-form on Σ).

(b) **Canonical choice of orientation** — There exists a smooth, nowhere-vanishing k-vector field $X \in \Gamma(\Lambda^k T\Sigma)$ that is distinguished by the geometry of Σ within M, in the sense that it is invariant under the full group of geometric symmetries preserving Σ as an embedded object.

The idea of orientable vs. non-orientable surfaces is a well-known one. But in our setting, we now require a slightly stronger definition for the helix, a "constantly-oriented manifold."

Example 2.4 (Rules for Geometric Quantization). Technically, any finite symmetry group of rotations of the circle, $C_n \subset SO(2)$, enables n-ary non-Boolean logic in geometric computability. For instance, one can define a 5-valued (pentary) logic using C_5—the cyclic group of order 5. More generally, we encode n-ary logic as follows.

Let $x \in \{0, 1, \ldots, n-1\}$ be the index of the rotation step, and m_x the associated logical value.

$$f : x \mapsto m_x, \quad m_x \in \mathcal{L}_n, \quad x \in \{0, 1, \ldots, n-1\}$$

where:

$$\mathcal{L}_n = \{\ell_0, \ell_1, \ldots, \ell_{n-1}\}$$

and the geometric correspondence is given by:

$$\theta_x = x \cdot \frac{2\pi}{n}$$

so the angle θ_x represents the logical value m_x.

Thus, a rotation of the helix by θ_x encodes the logical value ℓ_x, allowing n-ary logic representation via angular encoding. This mapping may be compactly written:

$$f(x) = m_x = \ell_x, \qquad \theta_x = x \cdot \frac{2\pi}{n}$$

Table visualization:

θ_x	Logical value
0	ℓ_0
$\frac{2\pi}{n}$	ℓ_1
$\vdots$	$\vdots$
$\frac{2\pi}{n}(n-1)$	ℓ_{n-1}

Definition 2.9 (Orientation Bit Assignment for n-ary Unit Helices and ω-logic)**.** Let the axis of symmetry of helix i be represented by a vector $\mathbf{v}_i = (v_x, v_y, v_z)$ in $\mathbb{R}^3$. We define a reference frame where the primary axis is $\hat{\mathbf{z}} = (0, 0, 1)$ and the reference direction for angle zero is $\hat{\mathbf{x}} = (1, 0, 0)$.

The n-ary logical state, or orientation bit $b_i \in \{0, 1, \ldots, n-1\}$, is determined by the angle of the vector's projection onto the xy-plane. This angle, θ_i, is calculated using the two-argument arctangent function:

$$\theta_i = \text{atan2}(v_y, v_x)$$

The arctan^2 function returns a value in the range $(-\pi, \pi]$. We map this to the range $[0, 2\pi)$ to ensure positive angles:

$$\theta_{\text{norm}} = \begin{cases} \theta_i & \text{if } \theta_i \geq 0 \\ \theta_i + 2\pi & \text{if } \theta_i < 0 \end{cases}$$

The continuous state (or fuzzy value) s_i of the geobit is defined as the normalized angle itself, representing a value in the real interval [0,1):

$$s_i := \frac{\theta_{norm}}{2\pi} \in [0, 1)$$

For n-ary discrete logic, the state b_i is derived by partitioning the continuous state space into n discrete bins:

$$b_i := \lfloor n \cdot s_i \rfloor = \left\lfloor n \cdot \frac{\theta_{norm}}{2\pi} \right\rfloor \in 0, 1, \ldots, n-1$$

This is termed "ω-logic."

Table 7 helps organize several models, showing the relationship between join symmetry and the resulting system of computation that comes as a result of it.

Table 7: The Chart of n-logics

Geometric Expressiveness	**Join Symmetry**	**Algebraic Closure Group**	**Result**
Extreme	0-fold	Diff(M) (Maximal)	Computational gauge field
High	1-fold (360°)	C_1	Geometric chaotic analog logic flow
Medium	2-fold (180°)	C_2	Classic boolean logic gate
Low	16-fold (22.5°)	C_{16}	Hexidemical logic gate
Minimal	n-fold (Infinite)	$SO(2)$ (Lie)	Algebraic fuzzy analog logic flow

This pattern reveals an astonishing phenomenon: at the extremes, logical computational flow manifests in two analog forms—one chaotically geometric, the other excruciatingly algebraic. This suggests that low-value n-ary logic, such as Boolean logic, constitutes a kind of *logical habitable zone* where real machines can function in physical space, much like how our own planet is the hospitable zone in our solar system.

It is not simply a quirk of engineering that noise and error rates make n-ary processors difficult to realize in practice. Even in an idealized, noiseless world, n-ary logical computers are "sandwiched" between two analog extremes.[5] The physically stable and computationally tractable zone is inherently narrow. Those regimes that are C_2 and above will be termed *algebraic*, and those that are below this threshold will be termed *geometric* computational regimes. [6]

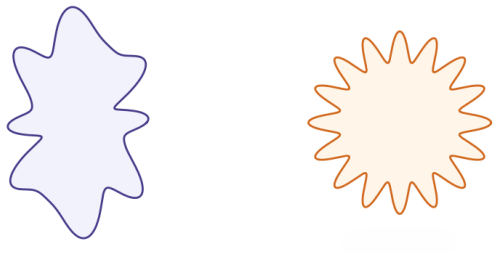
For n-ary logic, the cross section of the THM manifold can be any smooth closed manifold with at least n distinguished points on its boundary that realize the requisite logical "angularities." There are, of course, infinitely many such shapes. The primary family consists of sinusoidal perturbations ("clovers"), whose cross sections can be globally smooth and are easily visualized in polar coordinates—see Figure 3b.

However, one can readily imagine more expressive cross sections that realize the required angular structure but offer even richer forms of expressiveness. Figure 3a illustrates such alternatives and compares expressiveness across different logical regimes.

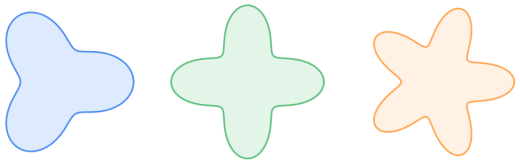
Definition 2.10 (Geometric Expressiveness)**.** Let $\mathcal{F}_1$ denote the space of all smooth, closed, simple curves (or hypersurfaces) in $\mathbb{R}^n$, parameterized up to some fixed regularity (e.g., finite-degree trigonometric polynomials, or a suitable Sobolev space). Let G_n be a symmetry group (e.g., the cyclic group C_n of order n), and let $\mathcal{F}_n \subseteq \mathcal{F}_1$ be the subspace of shapes invariant under the action of G_n.

[5] "Professor, why are noise rates from quaternary processors higher than binary?" "Ah, well, you see, the quantum universe is made of these glued together helices that we can rotate with electricity, but..."

[6] Please note a slight conflict regarding nomenclature, despite the fact that C_1 is an abelian group, its lack of an ability to define, at minimum, binary discreteness means it will fall under a non-abelian computational regime.



(a) Expressiveness in Binary vs Hexadecimal



(b) n-logical Gluing Maps for—3,4,5-ary

We define the *geometric expressiveness* at symmetry level n as the normalized dimension:

$$E(n) = \frac{\dim(\mathcal{F}_n)}{\dim(\mathcal{F}_1)},$$

where $\dim(\cdot)$ denotes the functional dimension (e.g., the number of independent Fourier modes up to a fixed cutoff, or the dimension of a relevant function space). Thus $E(n) \in [0, 1]$ quantifies the proportion of the full shape space available under the symmetry constraint.

Let $D(n, K)$ denote the dimension of the space of trigonometric polynomials of degree $\leq K$ and n-fold symmetry:

$$D(n, K) = 2 \left\lfloor \frac{K}{n} \right\rfloor + 1,$$

where the formula counts both cosine and sine harmonics, plus 1 for the constant. As n increases for fixed K, $D(n, K)$ decreases in a stepwise fashion.

For finite K,

$$E(n) \approx \frac{2\lfloor K/n \rfloor + 1}{2K + 1}.$$

The set of all 2π-periodic, smooth functions is infinite-dimensional. The subset of n-fold symmetric, smooth functions is also countably infinite-dimensional, but it becomes increasingly sparse as n increases. In the limit as $n \to \infty$, $E(n) \to 0$; the relative dimension shrinks to zero.

Furthermore, the total L^2 variation of $r(\theta)$ from a constant also decreases as n increases, since the only possible deviations are high-frequency harmonics, and these are increasingly penalized by the requirement of smoothness.

Definition 2.11 (Algebraic Verifiability)**.** Let $\mathcal{G}$ be a GCM, and let $O = \mathrm{Sym}(\mathcal{G})$ be the maximal group of algebraic symmetries acting faithfully on $\mathcal{G}$. The algebraic verifiability $\mathcal{V}(\mathcal{G})$ is a normalizable measure of the size and structure of O, representing the ability to distinguish and verify discrete computational states within $\mathcal{G}$. For a finite group,

$$\mathcal{V}(\mathcal{G}) = \log_2 |O|$$

A high $\mathcal{V}(\mathcal{G})$ corresponds to many, clearly distinguishable algebraic states (e.g., quaternary, hexadecimal), while a lower $\mathcal{V}(\mathcal{G})$ reflects fewer verifiable states or continuous/chaotic flows. For infinite or Lie groups, $\mathcal{V}(\mathcal{G})$ can be defined via the rank of a maximal discrete subgroup or another appropriate algebraic invariant (e.g., dimension of a maximal abelian subgroup).

Proposition 2.5. *The set of all closed, sufficiently smooth 1-manifolds in $\mathbb{R}^2$ reduces to the circle when subjected to arbitrary rotational symmetry constraints.*

Proof. A general sufficiently smooth closed curve in polar coordinates, which can be expressed as a Fourier series:

$$r(\theta) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\theta) + b_k \sin(k\theta)].$$

Suppose the curve is required to be invariant under rotation by $2\pi/n$ for some integer $n > 1$:

$$r(\theta + 2\pi/n) = r(\theta) \quad \forall\, \theta.$$

This constraint eliminates all terms except those where k is a multiple of n, so only terms $\cos(kn\theta)$ and $\sin(kn\theta)$ can remain for integer k.

As $n \to \infty$, the only nonzero term that can survive is the constant a_0. Therefore, in this limit, the curve reduces to:

$$r(\theta) = a_0,$$

so, under arbitrarily high-order rotational symmetry, the only possible closed 1-manifold in $\mathbb{R}^2$ is the circle. This proves Lemma 2.4. □

Remark 2.9. This is the essential phenomenon as the size of the rotational symmetry group grows. Once the symmetry group becomes continuous, the logic of the system becomes identical with its geometry. We stress that this is *not* merely a clever representation:

The computation and the manifold are outright one and the same.

This is the core idea of geometric computation, and it is why such systems can, in principle, exceed the computational power of a Turing machine—first by leveraging infinite algebra, then by accessing infinite geometry. The inevitable result is that "fuzzy logic" in this setting naturally leads to a computational state space governed by a compact simple Lie group—the ability to execute real-valued hypercomputation, not on an idealized machine, but a topologically stable medium.

Conjecture 2.6. *Manifold Expressiveness/Symmetry Tradeoff. Let G be a finite or compact Lie group acting on $\mathbb{R}^n$. Let $\mathcal{S}$ be the set of all smooth, closed hypersurfaces in $\mathbb{R}^n$ invariant under G. Then the functional dimension of $\mathcal{S}$ (in a suitable topology) is non-increasing as the order of G increases, and reduces to 1 (the sphere) for full rotational symmetry.*

This conjecture is a natural extension of the principle illustrated by explicit examples in this work to n dimensionality. Imposing higher degrees of symmetry through group actions systematically reduces the diversity of admissible geometries, ultimately constraining $\mathcal{S}$ to the family of n-spheres under maximal symmetry. While a complete proof is not presented here, it seems well within reach using established methods from representation theory and algebraic geometry. If established, such a result would clarify why only a restricted class of manifolds (such as those with limited symmetry) is suitable as GCMs.

Remark 2.10. Such a relationship is all well and elegant. But at this point, the reader may be questioning what precisely the point of "geometric expressiveness" is. Why not make all cross-sections a circle? The shape of the actual helices does not matter for classical computation insofar as the manifold remains smooth. This is a valid point, but it serves to illustrate that in the symbolic mode of operation, the true geometric versus algebraic tradeoffs cannot be fully realized. It is only as we move into the next steps of computation that the implications become clear. For now, it is taken as is.

It is now that we approach the bit space governed by chirality. While orientation in GCMs supports arbitrary n-valued logic through the structure of discrete rotation groups C_n, chirality remains fundamentally locked to ternary. The only values available are left, right, and a *degenerate unknown state* where torsion itself vanishes. Imagine if you will, a solid but smooth-ended toroidal elbow, a paradoxical helix with no twist. It simply extends outward for a short while from the glue joint and then upwards at a smooth angle. It is an inflection point between chiralities.

This explicit geometric sacrifice embodies Lemma 2.4 yet again. Greater logical expressiveness at the expense of geometric sharpness. The resulting manifold supports ternary logic in the chirality subspace. This is termed "χ-logic."

Even in higher dimensions, no known geometry allows for a richer chirality logic within a single connected manifold. Any attempt to construct such a system merely produces additional degenerate regions, not genuinely new logical values. The U position in the chiral space is a sort of "smooth singularity" [7] an inflection point between winding and unwinding. This asymmetry reflects the deeper topological distinction between angular (periodic) and chiral (binary with singularity) symmetries.

Remark 2.11. These changes, however, cause a shift in the overall group structure. The logical state space of the digit architecture can be generalized beyond binary logic by promoting the orientation bits to n-ary via the cyclic group C_n, and the chirality bits to ternary via C_3, representing left, right,

[7] I know this is imprecise, it is intended here to suggest a topologically regular but functionally critical transition point.

and degenerate states. The resulting maximal group structure then becomes $(C_n)^2 \times (C_3)^2$, but more importantly, it remains abelian, as all constituent groups and their product are abelian. This structure preserves the computational advantages of commutativity while vastly increasing the expressiveness of the geometric logic system. That way, these new "ternary $\times$ n-ary groups" also do not violate the previous laws we derived regarding both the abelian vs non-abelian digit space, and state space group embeddings.

Example 2.5. So if we so pleased, taking fully advantage of all the features we defined so far, we could construct a GCM that stored its n-ary digits under the structure:

$$|U014|\beta\rangle(\mathbb{Z}_{15})^2$$

and defined operations as such like:

$$|U014|(d_2)_B\rangle \xrightarrow{d_2} |U015|e\rangle$$

as an example operation to rotate unit helix B by 72°. The arity is such that chirality digits are ternary $(0, U, 1)$ and orientation digits are pentary $(0-5)$. This technically allows for the use of two different systems of logic in a single computational object, a computability process that can natively encode a *heterogeneous logical structure*.

A GCM is, at its core, not necessarily constrained by purely symbolically derived bits at all. It can model variable digit structures with extreme ease, but the exact number of bits in a GCM is itself a question of quantization. How exactly the overall geometric structures are quantized determines the accessible number of bits and, depending on how one chooses to quantize, the structure goes from completely discrete to completely analog. It is in effect a "bridge" between the discrete and analog computational worlds.

2.7 Syntax, Semantics, and Computation

It is at this point that we now execute a bold new synthesis that leverages three distinct but convergent lines of thought, in addition to our own.

First, starting with Mattias Felleisen's [9] theory of programming language expressiveness to provide a rigorous distinction between micro- and macro-expressiveness—the creation of new local states versus new global, compositional structures.

Second, Tai-Danae Bradley's [10] categorical semantics, which unifies the compositional structure of meaning in language and quantum information, provides the tools to model the meaning of computation as an emergent, compositional object: meanings are assembled from parts, but the whole can exhibit contextuality and entanglement—properties essential to macro-expressiveness.

Third, our own geometric and groupoid-based models turn these semantic notions into concrete objects: GCMs and decorated paths, where composition, context, and macrostate emergence become visible and manipulable. Here, macrostates are equivalence classes of geometric or computational microstates, and macro-expression corresponds to the ability to traverse or construct new geometric cobordisms.

Finally, by connecting these explicit geometric models to Baez and Dolan's cobordism hypothesis, we embed computation itself into the universal framework of extended TQFTs (see [11] for background, and how it plays out is in 10). Our constructions realize semantic tradeoffs—between expressiveness, verifiability, and tractability—as geometric and categorical phenomena. In this sense, we do not merely interpret computation using geometry, but construct the geometry of computation itself—linking the micro/macro-expressiveness of programming languages, the compositional semantics of meaning, and the universal classification of processes given by higher category theory. The following definitions will define the landscape of computational operational semantics:

Definition 2.12 (Micro-Expression)**.** A language extension is micro-expressive with respect to a base language if every program written using the extension can be mechanically translated (i.e., compiled or macro-expanded) into a program in the base language, without fundamentally changing the structure of the program. The extension is just "syntactic sugar"—it may make programs more convenient to write, but does not add true expressive power.

If for every program P in the extended language L', there exists a mechanically derivable program P_0 in L, such that the semantics of P and P_0 are equivalent, the extension is micro-expressive.

Definition 2.13 (Macro-Expression)**.** A language extension is macro-expressive (or just "expressive" in the sense of Felleisen/Sabry) if it allows programmers to write programs or express control/data abstractions that cannot be reduced mechanically or structurally to the base language without loss of power or change in semantic properties.

There exists at least one program P in L' for which no equivalent P_0 in L can be mechanically constructed, i.e., the semantics of P is not achievable in L alone.

Felleisen's foundational work on expressiveness in programming languages provides a rigorous semantic framework for distinguishing between micro-expressive and macro-expressive language features. They define precise criteria for when a language extension is merely syntactic (micro-expressive, or "eliminable by mechanical translation") versus when it adds genuine new expressive power (macro-expressive, or "semantically irreducible"). This distinction underlies much of modern programming language theory, providing the intellectual basis for our definitions of micro-expression and macro-expression, and by extension, for our broader semantic expressiveness tradeoffs in geometric and computational systems.

Just as in programming, where certain language features permit the creation of behaviors not reducible to simpler forms, in geometry, the gluing of submanifolds can create new topological invariants—if and only if the gluing is smooth (singularity-free).[8] As the complexity of the system increases, the possibility of singularities or obstructions to smooth composition rises, directly paralleling the limits of macro-expression in computation.

Definition 2.14 (Verifiability)**.** Let $\mathcal{S}$ be a computational or semantic system, with a set of properties or predicates $\mathcal{P}$ defined over its states or behaviors. The verifiability $\mathcal{V}(\mathcal{S})$ is a normalized measure of the system's capacity to algorithmically decide, prior to or independent of execution ("runtime"), the truth or falsity of nontrivial properties in $\mathcal{P}$.

$\mathcal{V}(\mathcal{S})$ is the (normalized) cardinality or dimension of the set of properties $p \in \mathcal{P}$ such that there exists an effective (algorithmic or algebraic) procedure to decide $p(s)$ for all relevant $s \in \mathcal{S}$, without recourse to brute-force execution or exhaustive enumeration.

In more general (semantic, geometric, or topological) settings: $\mathcal{V}(\mathcal{S})$ can be defined in terms of the number or structure of properties accessible to algorithmic, structural, or analytic verification (e.g., decidable invariants, checkable logical properties, or geometric signatures).

Verifiability is the system's capacity to algorithmically certify nontrivial properties of its states or behavior, prior to execution. In algebraic regimes, this can be measured by the size or rank of the symmetry group distinguishing states in semantic or geometric regimes. Verifiability reflects the set of properties or invariants accessible to algorithmic or structural verification. High verifiability corresponds to systems where many global or local properties can be decided by construction; low verifiability signals the presence of undecidable, chaotic, or non-algorithmic behavior. Micro-expression, the power to generate or describe new microstates by local extension, is often correlated with verifiability, but the two are not identical. Micro-expression is syntactic and generative, verifiability is algorithmic and semantic. In maximally micro-expressive systems, local behaviors are easy to define, but global, non-trivial properties may remain undecidable or unverifiable.

Here, Bradley's recent work in categorical quantum probability and compositional semantics describes a passage from classical to quantum probability—constructing quantum-like meaning and behavior from classical, combinatorial data via category theory. This mirrors, in the semantic realm, what geometric quantization does for physical systems: lifting classical structure to the quantum domain.

In our framework, the process occurs in reverse: we begin with deterministic, geometric models of computation—objects whose logic is embedded in their physical, topological structure. By analyzing the space of possible compositions, flows, and transformations, we find that the chaotic or macro-expressive regime gives rise to algebraic and probabilistic behaviors that are indistinguishable from quantum information theory. In other words, quantum structure emerges as the statistical or compositional shadow of geometric computation.

Definition 2.15 (Tractability)**.** Let $\mathcal{S}$ be a formal system (e.g., a programming language, a computational model, or a geometric computation system). Tractability is the property that there exists a procedure (algorithmic or semantic) to:

[8] For the physicists reading: this is not just a clever comparison. The logical geometric mismatches we have seen in the above section are precisely why—in the language of Feynman diagrams—certain gluings fail. The logic of the bordisms must agree.

- For machines: Execute or evaluate a statement/program/configuration in $\mathcal{S}$ with feasible computational resources (e.g., polynomial time or some agreed practical bound).
- For humans: Understand, reason about, or verify the intent and behavior of statements in $\mathcal{S}$ without overwhelming cognitive complexity.

More precisely: A property or decision problem in $\mathcal{S}$ is tractable if it can be decided or realized with computational resources (time, space, etc.) bounded by a polynomial (or similarly "practical") function of the input size. A system is tractable if most of its intended computations and the process of programming/understanding them fall within humanly or machine-manageable complexity.

Tractability, in our framework, refers to the ease with which both a machine can execute and a human can understand or reason about a computational statement in a formal system. Formally, a property or computation is tractable if there exists an effective procedure—bounded by practical resource limits (e.g., polynomial time, or manageable descriptive complexity)—to realize or verify it. For a programming language or system of symbols, tractability is both a computational and a social-epistemic phenomenon: the system acts as a Schelling point for shared meaning, enabling programmers (or agents) to coordinate understanding and reliably predict system behavior. The degree of tractability depends not only on algorithmic complexity, but on the alignment between the system's formal structure and how meaning is constructed and communicated among its users.

The categorical structure underlying these definitions will be explored *micro-expressively* in detail in future work. Here, we focus on explicit constructions, conjectures, and the motivation for the framework, to maximize accessibility and highlight the new conceptual terrain. So with the parts in place, we can now *macro-express* a new metatheorem:

Conjecture 2.7 (CORE THESIS: Expressiveness/Verifiability/Tractability ("EVT") Hypothesis). *The conjecture formally comprises two parts:*

(A) ***Unverifiability of Macro-expressive Systems.*** *Any system of computation supporting macro-expression is globally algorithmically unverifiable.*

(B) ***Expressiveness/Verifiability/Tractability Trilemma.*** *Within such a system, every deterministic computational model is fundamentally constrained by a three-way tradeoff among* expressiveness *(geometric or semantic generality),* verifiability *(algebraic checkability or formal runtime assurance), and* tractability *(algorithmic or computational efficiency). These attributes satisfy the following trilemma:*

No system can simultaneously maximize all three properties. For any given model, at most two of the three can be optimized beyond a critical threshold, while the third must necessarily be sacrificed.

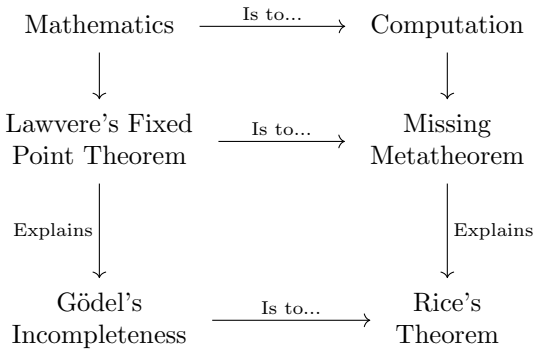
Formally, the achievable combinations of these properties are bounded by a 2-dimensional simplex in the space of attributes. The precise tradeoff is determined by the system's local symmetries and structural constraints.

This is the central motivating result of this work.[9] This is the core of our approach to understanding the nature of computation. It simply generalizes Lemma 2.4 to all forms of computation. We contend that there exists a deep connection here at work between verifiable ("algebraic") and expressive ("geometric") computational properties and that this applies to all deterministic computing systems. We also contend that, unlike conjectures such as the Church-Turing Thesis, this theorem is, in fact, provable via categorical construction, provided one makes explicit the categorical structures (e.g., cartesian closedness, enrichment, etc.) and operational definitions (e.g., what counts as "tractable"). To give a proper analogy, this is essentially a cousin of Lawvere's Fixed Point Theorem, but reapplied to computation at large.

The key to understanding lies in Rice's Theorem. This result is the natural consequence of the first half of the EVT Hypothesis. The EVT Hypothesis takes the core concept that Rice identified and situates it within a larger meta-framework that has so far been absent. I contend that there exists a deep and enduring parallel between Rice's Theorem and Gödel's Incompleteness. Rice's Theorem states that "All nontrivial semantic properties of machine languages are undecidable." Godel's incompleteness

[9] I understand Quantum Gravity and Yang-Mills are also important, and I can assure you this is still relevant to physics and comes in handy later.

states "There will always be statements within a formal system that are true but cannot be proven." Godel and Rice are two sides of the same coin—one for math, the other for computation. Yet this is not an immediately obvious connection. Godel only says that *certain* truths within a formal system may be true but unprovable, but most of math is still quite safe. But here, Rice is vastly more severe. *All* non-trivial properties are undecidable. If these two theorems are related, then why then the impedance mismatch between them?



While computer science has long known Rice's Theorem to be true, there has never been an explicit formulation as to *why* this is the case. Computer scientists acknowledge that many languages are more stable in their behavior than others. But no systemic attempt has been made to explain this, at least not in the same way that Lawvere did when he quietly saved mathematics by grappling with the paradoxes discovered by Godel and Tarski.

In mathematics, axiomatic structure contains undecidability to rare, specially constructed statements, so most of mathematics remains safe and verifiable. In computation, however, the absence of axiomatic constraints and the inherent flexibility (that is, "macro-expressiveness") of programming languages mean that undecidability is omnipresent. Inherent recursion means one is at the knife's edge of diagonalization at all times. This is the "impedance mismatch." Computation is vastly more "dangerous" than mathematics, because it allows unbounded self-reference and unfiltered logical construction, while mathematics relies on the ZFC to maintain order.

Hence, part of the problem lies with binary logic itself. Recall the chart of n-logics generated from join symmetry 7. Binary logic is extremely flexible when the spectrum of computational logics as a whole is considered. It teeters on the edge of chaos. One is a single false step into non-abelianism, though it provides ample opportunity for expression. If computers could express higher forms of n-ary logic easily, we might be able to reign in expressiveness with verifiability in hardware and constrain Rice's Theorem, but the laws of physics and the constraints of engineering decree this shall not be so!

But binary logic is not unique in its sins. Ternary or quaternary is nearly just as dangerous. One would have to go fairly high up the chain of n-ary logics to reach a logic restrictive enough to halt expressiveness to such a degree that you could avoid self-expression in computation and stop Rice's Theorem cold in its tracks. Normally, this is done with total recursion in proof assistant software, not hardware, using Coquand's Calculus of Constructions. [10] But wielding the power of type theory to its fullest potential means the computational process that one would be left dealing with would be limited to programming in a highly restrictive manner. In type theory, one has micro-expressiveness, but not macro-expressiveness.

Lawvere [12] identified that fixed points in category theory create endomaps which inevitably cause a slew of self-referential paradoxes. In the EVT Hypothesis, the fact we must confront is that the "fixed point" *is simply the computation itself*, or rather the ability for any system of computation to reproduce itself logically generating self-reference. Not every macro-expressive construct generates new fixed points, but most do. Turing made this assumption when he proved the halting problem. Turing's diagonalization in the proof of the halting problem is a powerful construction, but it is not a meta-theorem; it is a recipe. Unlike Lawvere's Theorem, which explained the inevitability of self-reference and undecidability in logic, there is no explicit meta-principle in computability theory that accounts for diagonalization in all computational systems. Felleisen, whether implicitly or explicitly,

[10] Imagine if you will, an n-ary geometric computer as being essentially a *hardware proof assistant*—it has infinite micro-expressiveness, so every state can be internally represented.

uncovered this when he argued for macro-expressiveness as the defining metric of expression in his seminal 1991 paper on the subject.

The EVT cleanly explains the odd "dual nature" one sees in functional programming languages. One side offers strict typing with meta-theoretical certainty during runtime, but meta-linguistic restrictiveness at compile time (Haskell). The other side, with dynamic typing, offers meta-linguistic expression in construction but no meta-theoretical certainty at runtime (Scheme). Attempts have been made to bridge the gap from both sides (MetaOcaml is to Coquand, as Typed Racket is to Felleisen), but if one uses too many features of these powerful languages, they can run into instability. **An explicitly stated conservation law for all forms of computational behavior explains these difficulties.**

The reason this meta-theorem has gone unformulated is that its essential tradeoff has long been hidden in plain sight—embedded in the very architecture of classical computation we all use in our day-to-day lives. The Von Neumann architecture, by treating program and data uniformly and allowing maximal flexibility (any memory can store code or data, and programs can modify themselves), enables *extraordinary* expressiveness. However, this very flexibility comes at the cost of global verifiability: it becomes almost impossible to reason about or prove properties of the system as a whole. When the hardware is instead fixed algebraically—restricting the set of possible states, operations, or transitions—the system gains predictability and becomes globally checkable, but loses expressive power.

But why do we say that the tradeoffs appear only at a certain limit as opposed being a fixed but constant quantity? Many computational processes are both low in expressiveness and low in verifiability. An example being the infamous "Turing Tarpit" (e.g., bash, Malbolge) where everything is possible, but doing anything is nontrivial. In the same way that Rice's Theorem must be harsher than Godel's Incompleteness to cover the whole range of computational behaviors, so too must my analogue to Lawvere's Fixed Points cover more conceptual ground when dealing with entire systems of logic. The issue is that Lawvere's construction on its own may not have the power to capture the issue of diagonalization within computation. He was focused on explaining the issues with mathematical logic, not computation as a whole. If computation is merely considered micro-expressiveness, then yes, Lawvere is sufficient for explaining the Halting Problem, but not its more generalized cousin—Rice's Theorem. Hence, I do not believe it to be conceptually complete about the full nature of computation as set out by Kleene and later Felleisen.

This is why now we, on the other hand, must grapple with two axes, not one. This is why we use only a quasi-Noetherian framing about local, not global symmetries, with particular computational models. In future work, I will publish my initial results on modeling this relationship in classical computing and drag it to light, but such work vastly outstrips the scope of this paper. The key difference for those readers wondering, however, is that in a Von Neumann machine, the most important local symmetry is within the skull of the computer programmer.

Remark 2.12. Regarding tractability, hardware is not truly a concern for geometric computation. If one had a stacked array of GLs, they could essentially form a hyperaware, omniscient FPGA capable of self-synthesis and logical introspection, switching between discrete and analog computational modes at will. In this regime, even the concept of a universal compiler or processor breaks down: every "program" defines its own hardware. In real life, there exist physical corollaries to Rice's Theorem and Lawvere's Fixed Point, natural limits on the engineering of computation itself in the form of finite, well-defined instruction set architectures, clear separation between compilation and execution, and static boundaries between code and metacode. As a result, although the entire trilemma must stated as-is, we will not be exploring tractability as much in this paper, as it is more of a physical constraint for humans and the machines we build, not the universe.

This is why the author will be truthful on the reasons for this model, and why this section is, in effect, the true introduction to this paper. In GCT, modeling Expressiveness/Verifiability relationships becomes much more tractable, since there is no separation between logic and memory, and tractability is built in, forcing tradeoffs between expressiveness and verifiability to the forefront. A GL is an idealized version of a non-Von Neumann architecture existing on a substrate defined purely by math—nothing but raw group theory, geometry, and topology—the true bare metal of our universe.[11] Here one can *see* how the operational semantics forcibly deforms the computational substrate, and model

[11] Dare I say—potentially the canonical one as a complement to the Turing Machine that has so far been missing for several decades.

how algebraic closure eats away at the possible expressions in geometry with increasingly verifiable but near infinite amounts of n-ary logic with every increasing bit! As Baez writes in *Getting to the Bottom of Noether's Theorem* [8]

> "Atiyah [13] warned us of the dangers of algebra: *" ...algebra is to the geometer what you might call the Faustian offer. As you know, Faust in Goethe's story was offered whatever he wanted (in his case the love of a beautiful woman), by the devil, in return for selling his soul. Algebra is the offer made by the devil to the mathematician. The devil says: I will give you this powerful machine, it will answer any question you like. All you need to do is give me your soul: give up geometry and you will have this marvellous machine.*"

Baez notes that "While this is sometimes true, algebra is more than a computational tool: it lets us express concepts in a very clear and distilled way." Although the striking resonance between these words and my argument may be coincidental, it is nonetheless unsettling. I do agree with Prof. Baez that the best of mathematics occurs when the two are brought together to work harmoniously, but my contention here is that computability is, unfortunately, one particular case of the bargain where the devil shall not be cheated. Under the watchful eye of the ZFC, algebra and geometry can play nicely together, but the act of computation (in our case—"gluing") creates a bottleneck. Though I must acknowledge that Atiyah himself likely never anticipated his words to be used in this way.

In this sense, the EVT Hypothesis may be viewed as a selection principle for the behavior of all computational processes. While it may have been originally inspired by the tradeoffs between programming languages, there is a deep pattern that computation—be it a flowing stream of geometry, or a flowing stream of electricity compiled from code—must respect hard limits on how much energy, information, and efficiency can be compressed into a single inch of space, and this is a limit imposed by our cosmos. Computation, be it discrete or analog, should not be viewed as an all-powerful, substrate-independent process, it is a gauge-constrained flow; embodied in both the user and the machine.

This core idea of *geometric constraint* is something that any physical theory of computation should respect, and this is why this paper is, at its heart, a physics one.

Remark 2.13. It is at this point I must pause, and sincerely apologize if this all may be a bit much for you, dear reader, to stomach. One does not open a paper with a whimsical title about helices expecting to receive a dialogue on identifying the deepest connections between math, logic, and computation, in such a soul-crushing manner, no less. If it is any consolation, the author shared similar feelings while writing this.

The convergence of categorical natural language theory and the abstractions of higher category and ∞-topos theory signals an emerging synthesis, and is a fitting reminder that mathematics, like language, is ultimately a science of meaning and structure, not just symbol and proof. Our foray into n-ary and fuzzy computation hints that the latent computational power within geometric objects has yet to be fully unlocked.

Let's turn the key.

3 Kleene Computation and Geometry

Stephen Kleene's work in the 1930s–40s established the modern theory of computability via the partial recursive functions, providing a machine-independent account that undergirds Turing machines and the Church–Turing thesis. In this section, we re-express GCT within Kleene's formalism. The point is twofold: (a) partiality, products, and groupoid symmetries appear naturally in our setting; (b) the presentation is platform-independent and does not commit us to a specific syntactic calculus (e.g., λ-calculus).

Our strategy is constructive. Concretely, we give semantics of primitive recursion inside our geometric category and then extend to partial maps (Kleene's μ-operator) in a way compatible with our groupoid symmetries. The categorical summary (via a Lawvere–theory view of primitive recursion and a partial-maps completion) will follow the operational definitions and examples below.

3.1 Fibered Kets and Product States

To track both observable and internal degrees of freedom of a geobit, we use a notational device reminiscent of the HV-ket. This is purely organizational: it packages a pair of states, one observable and one internal. When needed, we functorially linearize to a genuine Hilbert space; until then, the symbol $\otimes$ denotes the cartesian product of state spaces. While the HV-ket is useful for pedagogical purposes, a more advanced form of notation is now warranted.

Definition 3.1 (Fibered Ket for Geobits)**.** Let A be the (finite) observable state space encoding chirality and orientation, and let B be the internal state space (fluidic variables, or other hidden degrees of freedom). The *state object* is:

$$S \;:=\; A \times B.$$

A *fibered ket* is a notational pairing:

$$|\alpha\rangle \otimes |\beta\rangle \;\;\widehat{=}\;\; (\alpha, \beta) \in S,$$

where:

(a) $|\alpha\rangle \in A$ encodes the observable component (chirality/orientation).

(b) $|\beta\rangle \in B$ encodes the internal component (e.g. fluidic or topological data).

(c) the ambient state space S is an object of our geometric category $\mathcal{G}$, which is enriched by a groupoid of symmetries (see Section 4.7).

An elementary update (or local move) is a morphism $T : S \to S$, written on kets as

$$(|\alpha\rangle \otimes |\beta\rangle) \;\longmapsto\; |\alpha'\rangle \otimes |\beta'\rangle\,.$$

We write $G \triangleright T$ for the right action of an update T on a state $G \in S$ (so $G \triangleright T := T(G)$). When T factors by components, we write $T = T^{\alpha} \otimes T^{\beta}$ with $T^{\alpha} : A \to A$ and $T^{\beta} : B \to B$; otherwise T may couple the factors.

Remark 3.1. Throughout this section, the symbol $\otimes$ in the fibered ket notation is purely mnemonic and denotes the cartesian product of state spaces in our geometric category $\mathcal{G}$. No linear or inner-product structure is assumed, and no topological tensor product is used. If genuine tensor products are ever needed, they should be introduced explicitly; they play no role in the development of primitive recursion or partiality here.

Operationally, we will regard a GCM as an object subjected to computational flows: endomorphisms of S generated by finite vector fields or discrete moves. The recursor we define below iterates such updates, matching Kleene's scheme of composition and primitive recursion. In the categorical explanation that follows the examples, this will materialize as preservation of products and the natural-numbers-object (NNO) recursor in $\mathcal{G}$, with partiality handled by the partial-maps completion.

3.2 Partial Recursive Functions—Primitives and Composition

Classically, the partial recursive functions over $\mathbb{N}$ are generated from three primitives:

Zero $Z(\vec{x}) = 0$.
Successor $S(x) = x + 1$.
Projections $P_k^n(x_1, \ldots, x_n) = x_k$.

and three closures:

Composition given $f : \mathbb{N}^k \to \mathbb{N}$ and $g_i : \mathbb{N}^n \to \mathbb{N}$, define $h(\vec{x}) = f\big(g_1(\vec{x}), \ldots, g_k(\vec{x})\big)$.
Primitive recursion given $f : \mathbb{N}^m \to \mathbb{N}$ and $g : \mathbb{N}^{m+2} \to \mathbb{N}$, define $h : \mathbb{N}^{m+1} \to \mathbb{N}$ by

$$h(0, \vec{x}) = f(\vec{x}), \qquad h(n{+}1, \vec{x}) = g\big(n,\, h(n, \vec{x}),\, \vec{x}\big).$$

Unbounded μ–recursion given $g : \mathbb{N}^{n+1} \to \mathbb{N}$, define the partial map $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ by

$$f(\vec{x}) = \mu\, y \, \big[\, g(\vec{x}, y) = 0 \,\big],$$

i.e. the least y (if any) with $g(\vec{x}, y) = 0$.

We begin with the three primitive constructors of partial recursive functions. Let $(N_G, 0_G, S_G)$ be an NNO in $\mathcal{G}$; write N_G^n for products.

(a) **Zero:** For each arity n,

$$Z_n := !\,;\, 0_G \;:\; N_G^n \longrightarrow 1 \longrightarrow N_G,$$

the constant arrow to 0_G.

(b) **Successor:**

$$S_G \;:\; N_G \longrightarrow N_G.$$

(c) **Projections:**

$$\pi_k^n \;:\; N_G^n \longrightarrow N_G, \qquad \pi_k^n(x_1, \ldots, x_n) = x_k.$$

Example 3.1 (Zero as a Reset). Fix a readout $\mathrm{tw} : B \to N_G$ that extracts a discrete orientation count (or any integer-coded internal variable), and a setter $\mathrm{set0} : B \to B$ that overwrites this count with 0. Define the local move $T_0^\beta : B \to B$ by $T_0^\beta(\beta) = \mathrm{set0}(\beta)$ and act trivially on A. Then:

$$(|\alpha\rangle \otimes |\beta\rangle) \;\triangleright\; (\mathrm{id}_A \otimes T_0^\beta) \;=\; |\alpha\rangle \otimes |\,\mathrm{set0}(\beta)\,\rangle\,.$$

At the controller level, the associated numeric output is $Z_n(\vec{x}) = 0_G$, independent of inputs, matching the logical role of the zero function.

Example 3.2 (Successor as a Single Rotation). Let $\mathrm{tw} : B \to N_G$ be as above and let $\mathrm{inc} : B \to B$ implement $\omega(\mathrm{inc}(\beta)) = \omega(\beta) + 1$. Define $T_{+1}^\beta : B \to B$ by $T_{+1}^\beta = \mathrm{inc}$ and act trivially on A:

$$(|\alpha\rangle \otimes |\beta\rangle) \;\triangleright\; (\mathrm{id}_A \otimes T_{+1}^\beta) \;=\; |\alpha\rangle \otimes |\,\mathrm{inc}(\beta)\,\rangle\,.$$

On the numeric side, this is governed by $S_G : N_G \to N_G$, i.e. the successor on the orientation counter.

Example 3.3 (Projection as Readout/Selection). Given an input tuple $(x_1, \ldots, x_n) \in N_G^n$ serving as control parameters (e.g. planned numbers of twists, swaps, or steps), the projection π_k^n selects the kth control:

$$\pi_k^n(x_1, \ldots, x_n) = x_k.$$

Operationally, one can route $\pi_k^n(\vec{x})$ to an update family $\{T_y\}_{y \in N_G}$ to choose which local move to apply next; no change to the current geobit state is implied by projection itself.

Primitive Composition. Write $\pi_A : S \to A$ and $\pi_B : S \to B$ for projections. Fix a total map $v : B \to B$ (e.g. a one-step flow update).

$$\beta' \;:=\; v(\beta).$$

Define a discrete controller: $\sigma : B \to \{+1, -1\}$ by the sign of (some readout of) β':

$$\mathrm{sgn}(\beta') \;:=\; \begin{cases} +1, & \text{if the chosen readout of } \beta' \text{ is } > 0, \\ -1, & \text{otherwise.} \end{cases}$$

Any two-valued partition works; sign is a convenient choice. Observable update gated by the controller. Define:$f : A \times \{+1, -1\} \to A$ by

$$f(\alpha, +1) = \alpha, \qquad f(\alpha, -1) = -\alpha.$$

Then the observable component updates as:

$$\alpha' \;:=\; f\big(\,\pi_A(\alpha,\beta),\ \mathrm{sgn}(v(\pi_B(\alpha,\beta)))\,\big).$$

The whole composite step is the composite arrow:

$$T \;:=\; \big\langle\ f \circ \langle \pi_A,\ \mathrm{sgn} \circ v \circ \pi_B \rangle, \quad v \circ \pi_B \big\rangle \;:\; S \to S,$$

which is of the classical PRF composition form $f \circ \langle g_1, \ldots, g_k \rangle$.

Example 3.4 (Flow-controlled chirality flip). Take initial state $|\alpha\rangle \otimes |\beta\rangle = |{+1}\rangle \otimes |0.7\rangle$. Let $v(\beta) = \beta - 1$ (one-step slowdown). Then:

$$\beta' = v(0.7) = -0.3, \qquad \mathrm{sgn}(\beta') = -1,$$

then:

$$\alpha' = f(+1, -1) = -1.$$

Thus the updated state is $|{-1}\rangle \otimes |{-0.3}\rangle$.

Basic linear encoding. Choose a two-point basis for A: $|{+1}\rangle = (1, 0)$, $|{-1}\rangle = (0, 1)$, and a finite code for flow levels in B, e.g. $|\mathrm{low}\rangle = (1, 0, 0)$, $|\mathrm{med}\rangle = (0, 1, 0)$, $|\mathrm{high}\rangle = (0, 0, 1)$, together with a coarse-graining $q : B \to \{\mathrm{low,med,high}\}$ (so $q(0.7) = \mathrm{med}$, $q(-0.3) = \mathrm{low}$). Then the pair (α, β) is represented (after linearization) by the Kronecker product:

$$|{+1}\rangle \otimes |\mathrm{med}\rangle = (1, 0) \otimes (0, 1, 0) = (0, 1, 0,\ 0, 0, 0),$$

and updates act by block-permutation matrices implementing f on A and v (then q) on B.

Recursive composition is interpreted as primitive recursion over local moves. Fix an initializer $i : X \to S$ (the geometric constructor from inputs $\vec{x} \in X$), and a family of local moves $T_n : S \to S$ indexed by $n \in N_G$. Each T_n is an endomorphism of S (and in reversible regimes, an automorphism). We write right action as $G \triangleright T := T(G)$.

Define the state sequence by:

$$G_0(\vec{x}) = i(\vec{x}), \qquad G_{n+1}(\vec{x}) = G_n(\vec{x}) \triangleright T_n.$$

If T_n factorizes by components we write $T_n = T_n^\alpha \otimes T_n^\beta$; in general T_n may be a coupled update on $A \times B$. In fibered-ket notation:

$$|G_n\rangle = |\alpha_n\rangle \otimes |\beta_n\rangle\,, \qquad |G_{n+1}\rangle = (T_n^\alpha\,|\alpha_n\rangle) \otimes (T_n^\beta\,|\beta_n\rangle),$$

with the obvious coupled variant. After k steps:

$$|G_k\rangle = \big(\cdots((|G_0\rangle \triangleright T_0) \triangleright T_1) \cdots\big) \triangleright T_{k-1}.$$

Equivalence with the PRF recursor. Let $u : S \times N_G \to S$ be the step map $u(s, n) = T_n(s)$. The natural-numbers-object (NNO) recursor yields a unique:

$$h = \mathrm{rec}(i, u) : X \times N_G \to S$$

such that $h(\vec{x}, 0) = i(\vec{x})$ and $h(\vec{x}, n{+}1) = u(h(\vec{x}, n), n)$. Then for all k, $G_k(\vec{x}) = h(\vec{x}, k)$.

Example 3.5 (Commuting Flips on Separate Factors). Let $|\Psi_0\rangle = |\mathrm{left}\rangle \otimes |\mathrm{up}\rangle$. Let $C : A \to A$ swap left↔right and $O : B \to B$ flip up↔down. Define $T_0 = C \otimes I$ and $T_1 = I \otimes O$. Then:

$$|\Psi_1\rangle = (C \otimes I)\,|\Psi_0\rangle\,, \qquad |\Psi_2\rangle = (I \otimes O)\,|\Psi_1\rangle\,.$$

Hence:

$$|\Psi_2\rangle = (I \otimes O)(C \otimes I)\,|\Psi_0\rangle = (C \otimes O)\,|\Psi_0\rangle\,,$$

since C and O act on different factors and therefore commute.

Unbounded μ–recursion is interpreted as geometric search and partiality. We interpret Kleene's unbounded search inside the partial-maps completion $\mathsf{Par}(\mathcal{G})$. Fix a total $g : N_G^{n+1} \to N_G$ in $\mathcal{G}$ (a decidable numeric predicate with 0_G as "true"). The *partial* map:

$$\mu(g) : N_G^n \rightharpoonup N_G$$

is defined on $\vec{x}$ iff there exists y with $g(\vec{x}, y) = 0_G$, and then:

$$\mu(g)(\vec{x}) = \min\{\, y \in N_G \mid g(\vec{x}, y) = 0_G \,\}.$$

Constructively, $\mu(g)$ is realized as the increasing union of the finite-search partial maps $w_k : N_G^n \rightharpoonup N_G$ defined by primitive recursion:

$$w_0(\vec{x}) = \begin{cases} 0 & \text{if } g(\vec{x}, 0) = 0_G, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$w_{k+1}(\vec{x}) = \begin{cases} w_k(\vec{x}) & \text{if } w_k(\vec{x}) \text{ is defined,} \\ k{+}1 & \text{if } g(\vec{x}, k{+}1) = 0_G, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Then $\mu(g)$ is the partial map whose graph is $\bigcup_k \mathrm{graph}(w_k)$.

Geometric reading. For inputs $\vec{x} \in N_G^n$, an initializer $i : X \to S$ chooses a starting state, and a family of local moves $U_y : S \to S$ (or a step map $u : S \times N_G \to S$) generates candidates:

$$|G_y\rangle \;=\; U_y\big(i(\vec{x})\big).$$

A geometric halting predicate $H : S \to N_G$ (e.g. "symmetry satisfied", "entered special region") defines $g(\vec{x}, y) := H(|G_y\rangle)$. Then $\mu(g)(\vec{x})$ is the least y with $H(|G_y\rangle) = 0_G$ if such a y exists; otherwise the search is undefined at $\vec{x}$ (diverges).

Example 3.6 (Divergence via mutual triggering in a minimal two-site system). Consider two neighboring GCMs A and B with state factors $|\alpha_\bullet\rangle \in A$ (orientation: up/down) and $|\beta_\bullet\rangle \in B$ (continuous flow). The joint state is:

$$|\Psi\rangle \;=\; (|\alpha_A\rangle \otimes |\beta_A\rangle) \;\otimes\; (|\alpha_B\rangle \otimes |\beta_B\rangle) \;\in\; (A \times B) \times (A \times B).$$

Let $T : (A \times B) \times (A \times B) \to (A \times B) \times (A \times B)$ be the local transition that, at each step:

- increases β_A; if it crosses the threshold 1, flips α_A and triggers an increase of β_B;
- symmetrically for B (threshold 1, flip α_B, trigger A).

Set $|\Psi_{n+1}\rangle := T\,|\Psi_n\rangle$ and let the halting predicate be $H(|\Psi\rangle) = 0_G$ iff a designated stop condition holds (e.g. both flows = 0, or a marked symmetry cell reached). If T perpetually re-triggers the threshold crossings, then for all n, $H(|\Psi_n\rangle) \neq 0_G$ and the search never halts:

$$|\Psi_0\rangle \xrightarrow{T} |\Psi_1\rangle \xrightarrow{T} |\Psi_2\rangle \xrightarrow{T} \cdots$$

Hence for the corresponding $g(\vec{x}, y) := H(|\Psi_y\rangle)$ one has $\neg\exists y\, (g(\vec{x}, y) = 0_G)$, so $\mu(g)(\vec{x})$ is "undefined" as a partial arrow in $\mathsf{Par}(\mathcal{G})$. This realizes unbounded μ–recursion geometrically as an endless sequence of fibered-ket transitions with no fixed point.

3.3 Kleene Computability and Computational Groupoids

For a compact categorical definition to encapsulate our PRF correspondence, we turn to a Lawvere category.

Definition 3.2 (Lawvere Theory for Primitive Recursion)**.** Let LPR be the small cartesian category freely generated by an object N with arrows $0 : 1 \to N$, $S : N \to N$, finite products, and the primitive recursion operator characterized by the usual universal property.

Definition 3.3 (Geometric NNO)**.** In our semantic category $\mathcal{G}$ of geometric computations, a natural numbers object is an object N_G with maps $0_G : 1 \to N_G$, $S_G : N_G \to N_G$ such that for all X, $f : X \to N_G$, $g : X \times N_G \to N_G$ there exists a unique $\mathrm{rec}(f, g) : X \to N_G$ satisfying the recursion equations.

Theorem 3.1 (Representation of PRF)**.** *There exists a strict cartesian functor $F : \mathsf{LPR} \to \mathcal{G}$ with $F(N) = N_G$, $F(0) = 0_G$, $F(S) = S_G$, preserving products and primitive recursion. Consequently, every primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$ is realized by a unique morphism $F(\ulcorner f \urcorner) : N_G^k \to N_G$.*

Corollary 3.2. *Zero, successor, projections, composition, and primitive recursion are realized in $\mathcal{G}$.*

Let $\mathsf{Par}(\mathcal{G})$ denote the partial maps category over $\mathcal{G}$. It is cartesian, inherits N_G, and admits a (partial) unbounded search operator μ for decidable PR predicates. Thus:

Theorem 3.3 (Kleene Normal Form in $\mathsf{Par}(\mathcal{G})$)**.** *There exist primitive recursive T, U in $\mathcal{G}$ such that every partial recursive ϕ_e is represented as $\phi_e(\vec{x}) = U(\mu\, y.\, T(e, \vec{x}, y) = 0)$ in $\mathsf{Par}(\mathcal{G})$.*

Traditional models of computation rely on total functions or globally defined operations. But in the context of GCMs, computation is inherently local and context sensitive. Certain geometric transitions are only possible from specific states. This motivates a shift from group structures to groupoid categories, where every morphism is invertible, but only defined between certain object pairs. This is deeply analogous to Kleene's theory of PRFs, where computation is only defined on certain domains, and self-reference (recursion) manifests as groupoid automorphisms or loops, but embeds it within a reversible, topological framework. Thus, computation in a GCM becomes a path through a groupoid of geometric states, where the allowable morphisms reflect both physical constraints and computational structure.

Geobits are defined by observer-based phenomena. Observer-based phenomena require fixed points or planes. This should not come as a surprise. Kleene's Fixed Point Theorem [14] states that fixed points are a fundamental requirement for computation to exist. Information may indeed exist without the use of fixed points, but computation cannot. Computation by definition requires distinction, and all distinction requires a fixed frame of reference. This is merely a new and interesting way in which Kleene's theorem has manifested itself, albeit without the typical language of the lambda calculus and TMs. As we already have all PRFs defined, Kleene's Normal Form Theorem is automatically now in effect. We will tackle two theorems of Kleene.

Kleene's Recursion Theorem (the S-m-n Theorem): For every total computable function $\psi(e, x)$ (of two arguments: a program index e and input x), there exists a total computable function s such that

$$\forall e, x : \varphi_{s(e)}(x) = \psi(e, x)$$

That is, for every "program transformer" ψ, there is a computable way to produce a new program $s(e)$ that, on input x, computes exactly $\psi(e, x)$.

Kleene's recursion theorem is a fundamental result in computability theory. For any process that takes a program as input and produces a new program, there is always a program that can feed itself as input to that process. It is the foundational result for self-reference in computation. It is used in proofs of undecidability, the existence of viruses, quines, and in theoretical constructions throughout computer science.

Conjecture 3.4. *The Geometeric Recursion Theorem. Let F be any computable geometric transformation (e.g., a local automorphism or flow) that takes as input a description of a geometric state. Then there exists a state S^* such that*

$$F(\mathcal{E}(S^*)) = S^*$$

where $\mathcal{E}(S^*)$ encodes the relevant features of S^* itself.

Remark 3.2. If we assume there is a state encoding, then the geometric system can encode its configuration (group element, fibered ket state, parametric vector) as part of its own "internal data." We denote a geometric state as S, with an encoding $\mathcal{E}(S)$ that can be "fed" into a geometric transformation.

Consider a "geometric process" F that, given a description/encoding of a state, produces a new state (possibly a transformation or a new configuration):

$$F : \mathcal{E}(S) \mapsto S'$$

The goal then becomes to find a state S^* such that applying F to its description gives back itself (or a state that "acts as itself"):

$$F(\mathcal{E}(S^*)) = S^*$$

Creating a "geometric fixed point." Given that geometric computation encompasses a vast array of behaviors, this is not a trivial task.

Likewise, for Kleene's Fixed Point Theorem: For any total computable function $f : \mathbb{N} \to \mathbb{N}$ (mapping program indices to program indices), there exists an index e such that

$$\varphi_e = \varphi_{f(e)}$$

That is, the program with index e computes the same function as the program with index $f(e)$.

Conjecture 3.5. *Geometric Fixed Point Theorem. Let $\mathcal{M}$ be a GCM, equipped with a finite group $G \subset O(n)$ acting faithfully on $\mathcal{M}$, and let $\mathcal{S} \subset \mathcal{M}$ denote the discrete set of computational states stabilized under this action.*

Then there exists at least one point $p \in \mathcal{M}$ such that for every computable transformation $g \in G$, there exists a subgroup $H \leq G$ such that p is invariant under H:

$$\forall g \in G, \exists H \leq G \text{ such that } h \cdot p = p \quad \forall h \in H$$

Example 3.7 (Geometric Quine). Let $\mathcal{M}$ be a GCM with a finite group $G \subset O(n)$ acting faithfully on $\mathcal{M}$, and let $\mathcal{S}$ be the set of computational states stabilized under this action. Then for any computable transformation $g \in G$, there exists a configuration $p \in \mathcal{M}$ that is invariant under a nontrivial subgroup $H \leq G$; that is, $h \cdot p = p$ for all $h \in H$.

This configuration acts as a geometric quine, a self-replicating or fixed-point state for the corresponding computational action.

Remark 3.3. This conjecture and example are motivated by analogy with Kleene's fixed point theorem, where every computable "program transformer" admits a fixed point (a program that "feeds itself" to the transformer).

In the geometric setting, the group action $G \subset O(n)$ on $\mathcal{M}$ can be interpreted as a space of computable transformations or "automorphisms" of the configuration space. For each such transformation, the existence of a fixed point or stabilized configuration (i.e., a p invariant under a nontrivial subgroup H) is guaranteed by the finite nature of the group action and the full and faithfulness of G on $\mathcal{M}$, by a geometric analog of classical fixed point principles (think—Brouwer's Theorem), together with the combinatorial structure imposed by G.

The problem becomes that the search space of geometric states is *vastly richer* than in symbolic recursion. Geometric computation doesn't just include symbolic computation; it seeks to classify *all* deterministic computing. So there must be a property that some configurations are stabilized under nontrivial subgroups, and hence act as geometric "quines" or self-replicating states that must remain.

An exact proof of these two conjectures analogous to Kleene's classical results remains an open challenge. While the intuition and structure of self-reference persist in the geometric and groupoid setting, a fully rigorous construction encoding self-replicating states and transformations demands completely new categorical and geometric techniques. As well as doing so carefully concerning the computability and the topology of any $\mathcal{M}$, once collaborative refinement of categories in $\mathcal{ESM}^+$ is formalized. We've presented here a strong analogy and partial construction, but leave a full formal proof for future work.

4 Geometric Topology and Discrete Fluidic Computation

Recent work by Miranda et al. [15] on the computability of fluid flows—especially the Turing completeness of Euler's equations as conjectured by Tao [16]—has provided deep insight into the fluidic nature of computation. The challenge now is to realize these theoretical insights in a concrete geometric form.

Here, we lay the groundwork for understanding GCMs as geometric, quantum-like, topological computers. As established earlier, a GCM is neither a von Neumann machine nor a quantum computer. Quantum computers require coherence and impose strict limits on size, stability, and locality through entanglement. By contrast, one can increase the total geobit capacity of a geometric computer by chaining together GCMs into a lattice—the bits need not coexist within a single global state vector or collapse via quantum amplitudes.

In the following section, we demonstrate that such a system lies somewhere in between classical and quantum computation: deterministic, yet capable of far more than a traditional classical machine.

As a motivating example, consider the classic science experiment in which a hose filled with running water is placed across a vibrating speaker. The resulting shape of the water—observable as a visible, oscillating pattern—may offer insight into the principles underlying geometric computation. In this analogy, the speaker acts as the signal source or waveform generator (the "input"), while the hose and water jet serve as the computational substrate. The output is an incompressible fluid flow. The frequency of oscillation determines the tightness of the resulting coils, amplitude controls the radius of the spiral, and phase sets the initial offset or rotation. This observation demonstrates that computational geometric bordisms naturally manifest as smooth chiral manifolds. Helical and twisted structures arise as the minimal-energy configurations for storing twist, oscillation, and periodic driving under geometric and physical constraints. This principle is not merely theoretical: demonstration by brusspup [17] visually validates the emergence of smooth, helical manifolds in physical systems driven by periodic forcing.[12]

If it were possible to cross or interact streams of such geometric bordisms (as with THM structures), one could realize a physical lattice of GCMs.

Remark 4.1 (Computational Geometric Bordisms)*.* The vibrating hose experiment can be viewed as an analog of a *computational geometric bordism*, where the dynamic shape of the water from the hose mediates between different boundary conditions (e.g., fixed endpoints, or initial and final configurations). Here, the bordism is "internal" to the geometric evolution of the water. See section 5.3 for a breakdown of cobordism formalisms.

Remark 4.2*.* Recall definition 2.3. That result holds for a purely symbolic computational system. However, the true bit space easily encodable within a single THM under say C_4 ω-logic is more on the order of around 10 bits. This bound arises from additional parameters intrinsic to fluidic computation, such as:

Velocity — of fluid flow (e.g., a ternary "slow/medium/fast" trit.)
Direction — of fluid flow (binary "left/right.")

These fluid flows traverse along the screw axis of a unit helix—its "tape" to use a classical analogy. Thus, the geobit capacity grows quickly as more independent physical parameters are included. While it is theoretically possible to enlarge the state tuple $|\alpha\rangle$, the difficulty is that such variables are often "hidden"—their states are not directly observable, as are obvious geometric features like chirality and orientation. In symbolic regimes, these values could be masked but are ultimately "read" from the overall geobit value. In fluidic (physical) systems, they may become inaccessible or non-local.

Still, even in plain observables, helices provide ample opportunity for geometric quantization. The total number of twists could be interpreted as another trit in the $|\alpha\rangle$ of the geobit:

Helix Twists — of the unit helices ("low/medium/high"—corresponding to some defined table (e.g. "1-4/5-9/10-14") of the total τ.)

Proposition 4.1. *The geometric topology of a GCM encodes its information-theoretic structure, but it is also dependent on the subjective quantization of the model. Geometric computation is deterministic, but relativistic.*

[12] If the academic community will please excuse the author for using an unconventional reference to make a point about the nature of non-abelian gauge theory; I might also add that when one considers how Wilson loops and gauge invariance work, this isn't a bad teaching tool.

Unlike in classical computation—where bits are objective, absolute, and context-free—in geometric computation, the very notion of a "geobit" is a choice; it depends on how we partition the phase space, on which flows we treat as logic, and on the geometric substrate's topology. Geobits are fundamentally relativistic state structures. Their meaning and utility emerge only in relation to a chosen quantization or encoding scheme. I understand this invites criticism, but this is a feature, not a bug.

Just as in physics, where measurements and reference frames are part of the theory, so in geometric computation, information is a function of both the substrate and the encoding. The boundary between symbolic and fluidic regimes is thus not an objective fact about the system, but a reflection of how *we choose* to read and utilize its structure. When the shape and flow are the substrate, the encoding choice defines the computing model, and geometric deformation lets one interpolate between discrete *and* continuous logic.

This doesn't mean that we will not define the quasi-fluidic regime. In many ways, we already have. Our examples in Kleene use a hidden variable, a flowing field of vectors, to drive state behavior. This introduction of an explicitly defined information stream to drive behavior, even a hidden one, is the threshold that has been crossed. In addition, definition 4.2 allows us to retroactively understand the symbolic regime we have previously explored as that with which sits below our current phase space, in addition to the next fluidic phase of computation, without any redundant formalism.

4.1 Fluidic State Systems

We will now introduce the concept of a *Fluidic State System* ("FSS"). Semantically, this is done to show the non-discrete nature of GCMs. Classical FSMs cannot represent computations that depend on continuous parameters, boundary behaviors, or topological bifurcations —key features in geometric or fluidic computers. Computation at scale emerges from the coordinated action of many such FSS-governed units within the full lattice of GCMs. Thus, μ-recursion corresponds to a search not just within a single GCM, but within the broader manifold or lattice, as the system evolves according to local rules and seeks global halting conditions. So in the same way the finite state machine is the building block for classical computation, a fluidic state system becomes the building block for quasi-fluidic and fluidic modes in geometric computability.[13]

Definition 4.1 (Fluidic State System)**.** A FSS is a dynamical system defined by a smooth ambient manifold L equipped with a finite or countable set of distinguished regions $\mathcal{R}_i$, a family of continuous flows Φ_t on L governing its evolution, and a collection of transition interfaces $\mathcal{I}$ that map symbolic states to one another as the flow evolves. Inputs and outputs may be realized as boundary conditions, applied deformations, or perturbations of the flow field. Formally, an FSS is a 4-tuple:

$$(L, \{\mathcal{R}_i\}, \{\Phi_t\}, \mathcal{I})$$

Elaborated:

(a) **State** — L is the entire ambient lattice, and L_x in this case refers to an arbitrary subset of lattice cells connected by vector flows. Each cell of the lattice is a twin helix whose submanifold data ($\mathcal{R}_i$) — chirality (χ), orientation (ω), twists (τ), and global twin helix radius[14] (r) — encodes the local logical value; the global state is the list of all those values across the lattice.

(b) **Input** — External influence (Φ_t) enters as a smooth vector field applied to the ambient space. In practice, this could be mechanical torque, an acoustic wave, a laser pulse, etc. The vector field tells each unit helix how to start moving.

(c) **Transition** — As the input acts, the helices deform. Mathematically, this is a diffeomorphism of the manifold, a smooth reshuffling that obeys the governing differential equations (or, if one prefers the language of symmetry, a morphism in the relevant geometric groupoid). During this motion, the shape data (χ, ω, τ, r) may update, so the logical state changes. Each interface $I_{ij} \in \mathcal{I}$ is attached to a non-empty portion of the common boundary $\partial R_i \cap \partial R_j$ and consists of a smooth map

$$I_{ij} \ : \ \Sigma_i \times \Sigma_j \times \mathbb{R}^d \ \longrightarrow \ \Sigma_i \times \Sigma_j,$$

where Σ_i and Σ_j are the finite-dimensional state fibers carried by R_i and R_j, and $\mathbb{R}$ packages any local data extracted from the ambient flow (e.g. normal flux, shear, curvature). Operationally, I_{ij} is the update rule that

[13] In the future, the "Gauge" regime may require the formation of a "Topological State System."
[14] Note: individual radii of unit helices must remain equal for coherence reasons.

- Reads the current shape parameters (χ, ω, τ, r) on both sides.
- Samples the instantaneous vector field of Φ_t along the interface.
- Outputs the new shape parameters to be used after an infinitesimal time step.

Thus, $\mathcal{I}$ provides the "logic gates" that couple neighboring cells.

(d) **Output** — When the system settles, two things can signal a result. Either a new stable geometric configuration of the helices locking into a recognizable shape, or the emission of a secondary flow pattern, for example, a traveling pulse that can be read downstream. Either way, the final geometry carries the answer.

The conceptual map in Table 8 is provided for the ease of the reader.

Table 8: Classical vs. Quasi-Fluidic Computation

Classical Computation	**Geometric Computation - Quasi-Fluidic Mode**
Finite State Machine (FSM)	Fluidic State System (FSS)
Turing Machine (TM)	GCM Lattice (GL)
Bit	Geobit
State Transition Table	Groupoid Action + Local Flow Rules (ϕ, ϕ')
Tape Head	Local Update Regime $(\Phi(t))$
Output Tape	Observable Manifold Circuit $(\chi, \omega$ projection)
Alphabet Σ	Perturbation/Deformation Input Channel
Initial State s_0	Initial Geometric $\mathcal{R}_i$

Theorem 4.2. *FSS Geometric Embedding Theorem. In the quasi-fluidic mode of operation, each GCM admits an interpretation as a self-contained FSS fragment, with its internal and boundary structure encoding the computational components as part of its geometric computational logic.*

Proof. A soft proof is as follows. Let each GCM be modeled as a fibered state space L ($|\alpha\rangle \otimes |\beta\rangle$), equipped with a groupoid $\mathcal{G}$ of geometric transformations (twists, flows, orientation rotations).

Local evolution is governed by continuous flows Φ_t and discrete morphisms in $\mathcal{G}$, as in the 4-tuple definition. The distinguished regions $\mathcal{R}_i$ correspond to symbolic states, such as particular chiralities or orientation patterns (the "classical" logic states in a geobit). Transition interfaces $\mathcal{I}$ implement local update rules as morphisms in the groupoid, or, in fibered ket notation, operators acting on the tensor product structure. Inputs and outputs correspond to boundary conditions (incoming flow) and observations (projection via μ to a classical bit). Thus, for any classical FSM, one can construct a FSS out of a series of GCMs that emulates its transitions via a suitable arrangement of regions, flows, and interfaces. Conversely, every GCM in the quasi-fluidic regime can be described as a FSS, where the evolution is encoded in groupoid morphisms and the state is a fibered ket. The embedding is natural, as the groupoid captures all legal geometric "moves," and the fibered ket structure holds both the internal state and classical observables. □

Definition 4.2 (Quasi–Fluidic Computation and Regime Phases)**.** Given an FSS, we say a GCM operates in the *quasi–fluidic regime* if each region R_i carries a finite–dimensional state fiber:

$$\Sigma_i \;\cong\; B_i \times T_i \times Z_i,$$

with the following structure:

(QF1) **Discrete core (symbolic)** — B_i is finite, typically $B_i \cong (C_2)^{k_i}$; in the twin–helix unit one has $k_i = 4$ for chirality/orientation bits.

(QF2) **Phase layer** — T_i is a continuous but compact abelian Lie group (a torus), $T_i \cong (S^1)^{m_i} \simeq SO(2)^{m_i}$, encoding axial phase(s), orientation angles, or other bounded "analog" degrees of freedom.

(QF3) **Quantized invariants** — Z_i is a finitely generated abelian group (e.g. $\mathbb{Z}^{p_i}$), recording twist/winding/linking numbers or other topological charges.

(QF4) **Abelian symmetry and equivariance** — The local symmetry acting on Σ_i is

$$K_i \;\cong\; (C_3)^{k_i} \times (S^1)^{m_i},$$

and every interface map $I_{ij} \in \mathcal{I}$ and the induced flow on Σ_i are K–equivariant, that is, they depend on phase differences and commute with flips/rotations.

(QF5) **Flow coupling** — The ambient flow $\{\Phi_t\}$ transports the T–coordinates continuously (phase evolution) and can modify Z only by isolated, energy–threshold events (phase slips) prescribed by the interface rules; B updates by the same local rules. No non-abelian monodromy is present.

(QF6) **Non–singularity and bounded energy** — Trajectories have no finite–time blow-ups; curvature/torsion remain bounded; state updates at interfaces are well defined for all t.

A system is symbolic when $m_i = 0$ and Z_i is static; it is purely fluidic when B_i is trivial and non-abelian holonomy appears. The quasi–fluidic band sits between these: it combines a finite discrete core with $SO(2)$–type phases and integer topological counters, all governed by abelian symmetry.

4.2 Composition of Quasi-Fluidic GCMs in a GL

Previously, we assumed purely symbolic communication between lattices of GCMs, forming a GL. Even as we use quasi-fluidic computation, one can still assume a lattice model with symbolic communication between discrete units; it's simply that the discrete units now have much richer internal computational ability. But as we move further into quantum-like computation, a more fitting model is to view them as modules in a continuous topological flow circuit, each capable of taking in input via fluid, applying a transformation via its internal geometry, and outputting it to a connected "downstream" GCM.

Now, the potential path in this model composition is not adjacency in a single manifold but contiguity through flows. Each helix becomes a morphism from one flow regime to another. Their composition is a function composition: $H_3 = H_1 \circ H_2$. The discrete GCMs can willingly connect and disconnect as needed.

But how to direct the actual flows themselves? Recall that the chirality bits X in a geobit g are not particularly useful in terms of actually encoding information, being limited to ternary under C_3. But that is more than enough to serve for a different purpose—direction of flow. We use the following encoding table:

Chirality	Trit Value	Helix A	Helix B
L	0	Route Down	Route Left
NT	U	Halt/Disconnect	Halt/Disconnect
R	1	Route Left and Right	Route Right

This approach allows routing in 0 or 3 directions at once, forming a "computational topological flow circuit." The chirality of the unit helices themselves can encode 1 trit each. Just enough to communicate in a lattice. The U state of "No Twist" ("NT") represents "refusal"—neither input nor output. Hence, one could write code such as:

```
def fun untwist-unit-helix(unit){
    map undef → H[unit.χ]
}
```

To throw a halt on H_A in any given GCM while leaving H_B untouched.

Or to "reprogram" the actual topological flow of the computation similarly in the same way one can do in real life using an FPGA in Verilog or VHDL.

```
def fun untwist-even-row(row) {
let: column → ℕ
Where succ: ℕ → ℕ + 2
Foreach H in lattice[column,row]
    map: undef → H[a.χ,b.χ]
}
```

To globally set all even-numbered THM in a particular row into a "halt" state. The compiler would automatically generate machine code that would identify all even-numbered THM in said row and apply a global "untwist" operation. This is merely a "constructor" for setting up the initial state, though.

Ideally, we would like to route information in any direction across the lattice—up, down, left, or right. The full set of single- and double-directional flows is thus:

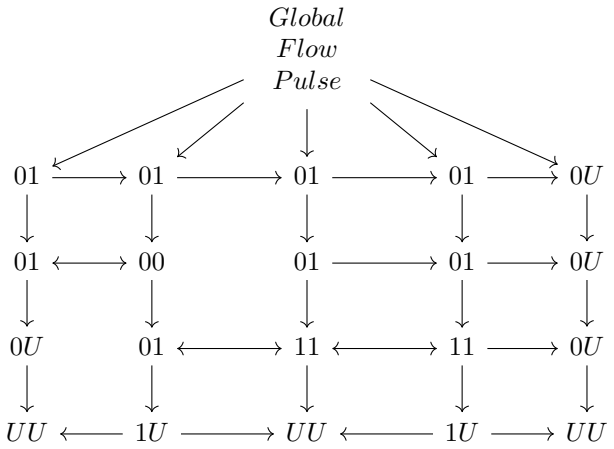
4 single-directional flows: {U}, {D}, {L}, {R}
6 two-directional flows: {U, D}, {U, L}, {U, R}, {D, L}, {D, R}, {L, R}
1 null direction: {}

This yields a total of 11 possible routing states. One might expect these to be encodable with two trits, but there is a crucial geometric obstruction: the "U" (undefined) state disconnects the logical model from the geometry. In a unit helix, setting the chirality bit to "undefined" degenerates the corresponding manifold segment, causing it to lose its twist, angular momentum, and, consequently, its computational capacity.

In the model developed here, chirality is intrinsically linked to the flow—there is no "undefined" chirality in a region with a well-defined direction of flow. Therefore, the resulting architecture must be a vertically-directed but horizontally-cyclic lattice.

Rather than a flaw, this "off switch" for chirality highlights a key feature of the underlying groupoid structure. Attempting to define an operation such as "increase the number of twists in unit helix H_i" while in the undefined state yields no morphism—there is no path in the groupoid connecting such states. Thus, the groupoid encodes both the allowed operations and the geometric impossibilities.

Despite these constraints, Turing completeness is preserved: loops can still be formed within the circuit, but are restricted to horizontal computation. This routing restriction simply reflects a deeper truth—certain logical states cannot be geometrically realized, and the groupoid structure naturally enforces this. An example of a computable circuit under these constraints is given below, along with the routing table.



The use of the U value is analogous to a Turing machine reaching a halt state. The direction of vector flows ceases at that location. However, unlike the Turing halt (which is terminal), the geometric system can be restarted by re-initializing the chirality, re-establishing the flow, and restoring the helix structure. Thus, undefined states act as reversible "pauses" in that segment of the computation, disconnecting the circuit locally, but not necessarily globally halting the entire process. True halt in GL only happens when the circuit *as a whole* is thrown in a disconnected state upon some globally defined end state as defined by the program. [15]

Remark 4.3 (Register Factorization in QF: Normal Forms and Tile Designs). Throughout this subsection, we work in the quasi–fluidic regime of Def. 4.2. For a given cell i, we write its state fiber as a product of the two unit–helix fibers:

$$\Sigma_i \;=\; \Sigma_{i,A} \times \Sigma_{i,B} \qquad \text{with} \qquad \Sigma_{\bullet,\star} \cong B_{\bullet,\star} \times T_{\bullet,\star} \times Z_{\bullet,\star},$$

and let $K \cong (C_2)^k \times (S^1)^m$ act K–equivariantly on each factor. Interfaces $I_{ij} \in \mathcal{I}$ are smooth, local, and K–equivariant.

[15] Again, the author would like to specify that this is his interpretation of geometric computation; my word is not law on the issue. Should one wish to use hexagonal or simplex arrangements and forgo this kind of chirality routing, they are free to do so.

Proposition 4.3 (QF Register Normal Form)**.** *In the QF regime, for each cell there exists a local K–equivariant change of coordinates on $\Sigma_{i,A} \times \Sigma_{i,B}$ such that every incident interface map I_{ij} is block–triangular with respect to the A/B split. Concretely, after reparametrization one has, for each neighbor j,*

$$I_{ij} \;=\; \begin{pmatrix} F_{AA} & 0 \\ F_{BA} & F_{BB} \end{pmatrix} \quad or \quad \begin{pmatrix} F_{AA} & F_{AB} \\ 0 & F_{BB} \end{pmatrix}.$$

By design (setting the off–diagonal block to 0), one obtains independent registers per helix. *Meaning the cell is* independent *if all off–diagonal blocks vanish on all incident interfaces; it is* glued *otherwise.*

Proof sketch. Abelian symmetry and K–equivariance restrict interface maps to commute with a common diagonal action on the T–phases and flips on B. Locally, one may choose K–adapted coordinates that separate control and data directions; because the allowed operations on the two helices commute, the interface family can be simultaneously put in a block–triangular form. Moreover, since interfaces are engineered artifacts in QF, the off–diagonal block can be set to 0 by construction, yielding decoupled per–helix registers. □

Definition 4.3 (Twin Helix Coupling Matrix)**.** For a cell i, the *coupling matrix* C_i is the collection of its off–diagonal interface blocks across all neighbors,

$$C_i \;:=\; \{\, F_{AB}^{(ij)} \,,\, F_{BA}^{(ij)} \mid j \sim i \,\}.$$

We call i *separable* if $C_i = \varnothing$ (all off–diagonal blocks vanish), and *coupled (glued)* otherwise.

The normal form above leads to two canonical tile choices; both remain within QF and preserve K–equivariance.

Definition 4.4 (RNF-1: Single-Register Tile)**.** Designate H_c (helix A) as a *control/router* carrying χ (and only minimal T/Z as needed), and H_d (helix B) as the *data register* carrying ω (and optional Z counters). Interfaces are chosen upper–triangular,

$$I_{ij} \;=\; \begin{pmatrix} F_{cc} & F_{cd} \\ 0 & F_{dd} \end{pmatrix},$$

so that data does not back–drive control. Each cell thus realizes one computational register with an internal control line.

Definition 4.5 (RNF-2: Dual-Register Tile)**.** Keep both helix–registers as data–bearing and declare the coupling matrix C_i. Tiles with $C_i = \varnothing$ are *separable* (two independent registers); tiles with $C_i \neq \varnothing$ are *coupled (glued)* and must document which QF axioms and K–equivariant constraints permit the specific cross–influences F_{AB} or F_{BA}.

Remark 4.4 (Default and Compilation). For clarity and predictable routing, we adopt RNF–1 as the default tile (single register per cell, control via χ, data via ω). High–density designs may switch to RNF–2, but must specify C_i explicitly. Either choice is compatible with the universality criterion of 4.4: the discrete computation resides in the B–layer, while $T \times Z$ act as controlled carriers.

4.3 Well-Behaved Vector Fields in $\mathcal{ESM}^+$ and the Paradox of Analog Computation

The categorical space $\mathcal{ESM}^+$ ("Embedded Smooth Manifolds with Enrichments") will serve as the natural setting for all GCMs, as explained later in Section 5.4. Moving beyond purely symbolic computation, we must now grapple with the mathematics of vector fields.

One of the key motivations behind GCT is to avoid the classical reliance on PDEs for modeling computation over continuous media, such as fluids or fields. As Prof. Tao has repeatedly warned us, PDEs introduce complexities that can develop singularities and "blow up" or lose regularity.

The fluidic integrators of the past and the analog computers used for World War II artillery tables are examples of how humanity has exploited the computational properties of fluids, while also encountering their mathematical hazards. While many analog machines merely simulated digital computers hydraulically, some fluidic computers (such as Soviet liquid PDE integrators) performed actual analog computation. But herein lies the paradox: the computational substrate is not immune to the pathologies of the mathematics it enacts.

One could compute the behavior of liquids, but only by using liquids themselves. If a liquid PDE solver were poorly engineered or maintained, singularities within the fluid could ruin the computation. Likewise, if the equation being computed contained singularities, the liquid would misbehave. The medium was the message, and vice versa. Unlike digital computers, which tend to fail noisily, analog computation can collapse silently.

GCT circumvents this by introducing local, deterministic transition rules that enable computability by structure rather than by solving equations. As a result, GCMs are inherently piecewise differentiable. We do not intend to solve Navier–Stokes or similar systems by computing bit-flips or logical steps. Instead, we seek to geometrically encode logic into the shape and interaction of embedded manifolds. In a GL, the computer is the problem, and the problem is the solution.

It should be noted that this approach is not without precedent. Coincidentally, Alain Goriely—whose earlier work on nonlinear geometry and helices will be cited later—has recently advocated a similar philosophical shift from global PDE-based approaches towards discrete, network-based modeling in neuroscience. His group's recent discrete brain modeling work demonstrates clearly how complex, globally emergent behaviors can be successfully captured by purely local computational rules, mirroring an approach to replace global PDEs. See Thompson et. al. [18] for a foundational implementation on understanding neurodegenerative diseases with networked systems of ODEs.

The following are the constraints for "well-behaved" vector fields in a GCM:

(a) **Continuity** — The vector field $\vec{F}(x)$ must have no sudden jumps or discontinuities.

(b) **Differentiability** — $\vec{F}(x)$ must be at least C^1.

(c) **Lipschitz Condition** — The field must satisfy a Lipschitz condition to ensure well-posedness of solutions to $\frac{d\vec{x}}{dt} = \vec{F}(\vec{x})$, preventing non-deterministic behavior due to multiple trajectories from a single point.

(d) **Boundedness or Decay at Infinity** — $\|\vec{F}(\vec{x})\| \to 0$ as $\|\vec{x}\| \to \infty$.

(e) **Tangency to Boundary** — The vector field must remain tangent to any topological boundaries—never "jumping off" into space, but lying flat along surfaces.

Each of these conditions helps address the potential blind spots of the others, ensuring a robust, multi-pronged approach to the smooth flow of fields over smooth manifolds.

4.4 A Dynamical Systems Approach for Modeling FSS with ODEs

At this point, we firmly enter the realm of dynamical systems modeling to understand fluidic computation in a GL. It is not our intention to use differential equations to define the model; rather, we use them to describe or analyze the computational dynamics when flow is involved. That is, we are modeling how fluid flow perturbs a helix's radius, twist, and so on, and wish to write down a few governing equations. These are not fundamental to the notion of geometric computability—they are interpretive tools, much like how logic circuits can be analyzed using Ohm's Law or other equations in electronics, but the logic itself does not depend on such analysis.

The following is a toy model that serves to deepen our understanding of quasi-fluidic computation.

Definition 4.6 (The Fluidic Helix Computational Model ("FHCM"))**.** The FHCM is an example of a dynamical system that, when sufficiently composed, can simulate the computation of helices in a GL. We define a helix $\mathcal{H}(t)$ embedded in $\mathbb{R}^n$, where t is a parameter along the helix (arc length or time). Each unit helix is equipped with the following dynamical state variables:

(a) **Chirality** $\chi \in \{+1, 0, -1\}$ — right-handed, left-handed, or undefined. Governs the direction of flow and computation.

(b) **Orientation** $\omega \in \left\{\frac{2\pi k}{N} : k = 0, \ldots, N-1\right\} \cong C_N$ (or, in the fully analog case, the continuous circle S^1). The orientation state space can be identified with a cyclic group of order N; the value of N encodes the logical base and simultaneously determines the 2^n state capacity. The underlying rotation logic sets the cross-sectional geometry.

(c) **Flow Rate** $\phi(t) \in \mathbb{R}$ — fluid speed along the helix screw axis. A generalized hidden variable that can drive state changes. Quantized at the user's discretion.

(d) **Flow Direction** $\phi'(t) \in \{+1, 0, -1\}$ — the sign of the derivative, indicating rightward (+), halt (0), or leftward (-) flow. Sets the local chirality. While technically optional, it is fundamentally just $\frac{d\phi}{dt}$ and so is naturally included.

These represent the core variables in the quasi-fluidic regime. Additionally, we may include:

(a) **Radius** $r(t) \in \mathbb{R}^+$ — the scale of the helix. Larger shapes are analogous to higher memory bandwidth. Optional for core computation, but can represent "scale of computation."

(b) **Twist Density** $\tau(t) = \frac{d\phi}{dt}$ — angular velocity around the axis, acting as a "clock rate." Higher twist means faster iteration through states.

(c) **Tube Volume** $V(t)$ — can be seen as analogous to data throughput, or more aptly, parallelism. Greater volume allows for more flows.

(d) **Phase Offset** $\theta_{\min} \in [0, 2\pi)$ — the starting point in the twist cycle. It can serve as an "initial state" or "addressing" mechanism.

(e) **Length** $\mathcal{L}$ — physical length of the helix, determining the extent of the computation cycle or memory span. In a finite system, this bounds computational depth or state space.

These additional variables are not strictly necessary for modeling computation in this regime, but when modeling under actual physical or analog constraints, they become useful to consider. In geometric computation, the user is empowered to define the model to their needs: the more of these variables included, the closer one comes to the full fluidic regime, even if you only cross into this state upon breaking algebraic closure and descending to C_1 under the rotation group.

How then does computation work in FHCM? A computation step is modeled as a global pulse causing a local change in $\phi(t)$ as dictated by each GCM transmitted across a GL. Core functions are as follows:

Flow direction causes chirality swaps — If $\phi'(t) > 0$, swap $\chi_0 \to +\chi$ of H_i. If $\phi'(t) < 0$, swap $\chi_0 \to -\chi$ of H_i. If $\phi(t) = 0$ set $\chi = 0$. As a table mapping for each unit helix H_i:

$$\chi_i = \begin{cases} +\chi & \text{if } \phi'(t) > 0 \\ -\chi & \text{if } \phi'(t) < 0 \\ 0 & \text{if } \phi(t) = 0 \end{cases}$$

Orientation rotates relative to flow rate — A canonical form can be given by a user-specified bijection

$$F : \mathbb{R} \longrightarrow \mathbb{Z}/N\mathbb{Z}$$

or, for practical computation, Choose N intervals in $\phi(t)$, one for each orientation state. The piecewise function becomes:

$$\omega_i = \begin{cases} 0 & \text{if } \phi(t) \in I_0 \\ 1 & \text{if } \phi(t) \in I_1 \\ \vdots & \\ N-1 & \text{if } \phi(t) \in I_{N-1} \end{cases}$$

Or for use of analog logic flow (e.g. $SO(2)$):

$$\omega_i = G(\phi(t)), \quad G : \mathbb{R} \to S^1$$

Here, G could be any continuous, monotonic, or user-specified function mapping flow rate to orientation angle.

And behavior for the optional extensions:

Radius growth = bandwidth expansion — $r(t)_x \to r(t)_{x+1}$ indicates increased capacity— parallelism or memory.
Twist density increases with frequency — More twists is equivalent to a higher computational "clock rate."
Phase coupling — Helices align their θ_{min} if flow is synchronized (coupled oscillators).
Length — The physical length of the helix grows or contracts as a function of twist density and radius, reflecting the ability to scale up computation or memory.

The hidden variables $\{\phi, \phi'\}$ are especially useful for our purposes, as there is a clean mapping between their values and the orientation and chirality state variables $\{\omega, \chi\}$.

The main logic operations are implemented by orientation rotations of varying density. Although it is physically possible to model sequential transitions between states (e.g., $0° \to 90° \to 180°$), the true computational power of the model arises from the ability to transition directly between any pair of allowed orientation states in a single operation, provided the flow pulse is configured appropriately. **This allows multi-bit logic flips**—analogous to parallel processing in digital systems—and is a key source of potential speedup in geometric computation (see Section 4.10).

Chirality is mapped in a discrete, one-to-one correspondence with $\phi'(t)$. Flow in this context refers to two local flows on a single manifold, one for each unit helix, and is not necessarily a global field. Importantly, zero flow is not an "absent" state; as previously explained, it represents a local halting condition. Thus, computation can proceed globally, locally, or be paused in any region of the lattice, fully mirroring the flexible control of both digital and analog logic.

Both binary and multi-valued logic are possible: the orientation state ω may take values in C_N (i.e., $\mathbb{Z}/N\mathbb{Z}$), or for the binary case, just two values. In the binary regime, standard Boolean gates (AND, OR, NOT, etc.) are implemented as local transitions triggered by flow pulses. In the generalized (n-ary) case, the same mechanism extends naturally to multi-valued logic operations, with orientation rotations corresponding to cyclic logic gates. The model is thus highly flexible: it recovers classical digital logic as a special case (C_2), while supporting richer logics (C_4, C_∞) for advanced forms of computation. The examples below are presented under binary ω-logic.

(a) **AND/OR gates** — Both input THM have their chirality set to route into the downstream THM, whose internal transition rules encode the relevant truth tables for the operations. A flow pulse then executes the operation.

(b) **NOT gate** — One input THM routes to a downstream THM, and it automatically inverts the given orientation on a flow pulse.

(c) **Memory register** — Configuration is stable as long as chirality and orientation remain unchanged and no flow pulse disrupts it.

(d) **Clock** — Global periodic flow pulse triggers all logic transitions per chirality routing at that tick

Note regarding physical input limits. Each node in the lattice can receive flows from at most three directions $\{D, L, R\}$, setting the maximum in-degree of the graph as 3. But internal logic arity in ω can be in any one of N possible states, not just ternary. Logic of the nodes in the graph as a whole can be arbitrarily rich, determined by the structure of ω (e.g., for quaternary logic, $N = 4$, for analog logic flow, N is infinite). Hence, the node's update rule can compute any function:

$$f : \mathbb{Z}/N\mathbb{Z}^3 \to \mathbb{Z}/N\mathbb{Z}$$

This is why we use the term "cyclic logic gate."

Definition 4.7 (Cyclic Logic Gate)**.** Let $N \in \mathbb{N}$ with $N \geq 2$. A *cyclic logic gate* of arity k is a function

$$f : (C_N)^k \to C_N,$$

where $C_N \cong \mathbb{Z}/N\mathbb{Z}$ is the cyclic group of order N.

In the geometric computation context, each node in the lattice receives up to k incoming "flow" states, each taking a value in C_N, and computes a new output state via f. The update rule for each node is thus specified by such a function.

For example, when $k = 3$ (corresponding to directions $\{D, L, R\}$), the node's transition is given by

$$f : (C_N)^3 \to C_N.$$

If $N = 2$, f is a Boolean logic gate; if $N > 2$, f is a multi-valued cyclic gate. For $N \to \infty$, f is an analog gate over S^1.

Remark 4.5. Unlike traditional n-ary logic gates, where both the number of inputs and the logic alphabet are fixed (e.g., a ternary gate maps $(C_3)^3 \to C_3$), the cyclic logic gate in this geometric framework is defined by a physically fixed arity (k, set by the lattice's connectivity) but allows the logic alphabet n to be arbitrarily large. Thus, each node processes up to three inputs but may output any n-valued state. In GCT, each wire can be its own algebra—thanks to the helix.

Example 4.1 (Helix Evolution Equations as ODEs). We describe the evolution of the optional physical variables through local coupling between geometry and flow, modeled by the following ODEs:

Radius Growth — Bandwidth/Parallelism:

$$\frac{dr}{dt} = \lambda_r \cdot f_r(\text{load}, \phi(t))$$

Here, the radius r increases in response to higher computational load or flow rate. λ_r is a growth rate constant, and f_r is a user-chosen function—possibly as simple as $f_r = \phi(t)$, or based on local memory pressure.

Twist Density — Clock Rate:

$$\frac{d\tau}{dt} = \lambda_\tau \cdot g_\tau(\text{signal}, t)$$

The twist density τ increases in response to a global or local clock signal, or as a function of time. λ_τ is a coupling constant; g_τ could be a pulse train or feedback from global clocking.

Phase Coupling — Synchronization:

$$\frac{d\theta_i}{dt} = \Omega + K \sum_{j \sim i} \sin(\theta_j - \theta_i)$$

This is the standard Kuramoto model [19] for coupled oscillators. Each helix i aligns its phase θ_i with its neighbors, with coupling strength K and natural frequency Ω. While the quasi-fluidic GCM model is globally synchronous by default, local helices can synchronize via phase coupling. This is analogous to asynchronous Muller C-gates, enabling emergent timing behavior within the lattice.

Length Evolution — Extent of Computation:

$$\frac{d\mathcal{L}}{dt} = \lambda_L \cdot h_L(\tau(t), r(t))$$

The physical length $\mathcal{L}$ of the helix grows or contracts as a function of twist density and radius, reflecting the system's ability to scale up computation or memory. λ_L is a scaling constant; h_L may be proportional to $\tau(t)r(t)$ or follow any other chosen law.

Remark 4.6 (Why Helices?). Helices are the simplest, omnipresent carriers of chirality and twist. They encode a vector (orientation), a phase (turns), and a topological charge (linking number) in one stroke. Using them lets us replace abstract real registers with physical, finite-energy geometry and, in doing so, marry analog power with fault tolerance. While we originally used the term "helical solid" in the beginning of this paper, the reader should visualize the helix as a "symbolic flow manifold." The direction in which it flows *is* the logic. Though we must stress that these are merely interpretive tools, and that more advanced computational approaches are likely warranted in future work. The core elements have been defined explicitly above; the extension variables, which are more physical and continuous, are intentionally left as differential equations.

4.5 Universality and Non–Universality in Cyclic Logic

We now make precise when the quasi–fluidic layer recovers classical universality and when it provably falls short. Throughout, assume the QF hypotheses of Definition 4.2 (finite discrete core B, abelian symmetry, K–equivariance, bounded energy, no blow–ups) and the FSS framework $(L, \{\mathcal{R}_i\}, \{\Phi_t\}, \mathcal{I})$.

Definition 4.8 (Pulse, Return Map, and Clopen readout)**.** A global pulse is a time–Δt drive that induces the map $\Phi_{\Delta t} : L \to L$. Let $\mathcal{P} = \{P_a\}_{a \in B}$ be a finite clopen partition of L induced by the interfaces in the sense that for every cell R_i there is a surjection $\pi_i : R_i \twoheadrightarrow B$ with $R_i = \bigsqcup_{a \in B} \pi_i^{-1}(a)$ and such that all $I_{ij} \in \mathcal{I}$ are K–equivariant and respect $\mathcal{P}$. The discrete readout of a global state $x \in L$ is the configuration $b = (b_i)_{i \in \mathbb{Z}^d} \in B^{\mathbb{Z}^d}$ defined by $b_i = \pi_i(x|_{R_i})$. The return map on B is the induced update:

$$\mathcal{F} \;:=\; \text{Read} \circ \Phi_{\Delta t} \circ \text{Encode} \;:\; B^{\mathbb{Z}^d} \longrightarrow B^{\mathbb{Z}^d},$$

where Encode places the B–labels as representatives in L cellwise.

Theorem 4.4 (QF Universality Criterion—Discrete Core)**.** *Suppose there exists a finite clopen partition $\mathcal{P}$ such that the return map $\mathcal{F}$ is local and shift–commuting; equivalently, there is a finite neighborhood $\mathcal{N} \subset \mathbb{Z}^d$ and a rule $f : B^{\mathcal{N}} \to B$ with*

$$(\mathcal{F}(b))_i \;=\; f\big(b_{i+\mathcal{N}}\big) \quad \text{for all } i \in \mathbb{Z}^d.$$

Then the GL realizes a cellular automaton on the alphabet B. In particular, by compiling a known universal CA local rule into the interface family $\mathcal{I}$ (using the QF routing and K–equivariant updates), the GL simulates a Turing machine.

Proof sketch. K–equivariance and the clopen partition ensure that a pulse transports each cell's microstate within its P_a–block according to data from a finite neighborhood (interfaces are local). Thus $\mathcal{F}$ is a CA global map on $B^{\mathbb{Z}^d}$. Universality follows by instantiating f to a universal CA rule and compiling its local table into the I_{ij}; chirality–based routing supplies the required wiring. The continuous $T \times Z$ coordinates act only as controlled carriers; the computation occurs on B. □

Remark 4.7 (Design Corollary)*.* Practically, one chooses $\mathcal{P}$ by thresholding the T–phases and fixing Z within energy bounds, so each pulse is a synchronous nearest–neighbor update on B. The construction is compatible with the QF constraints (abelian symmetry, bounded curvature/torsion).

Proposition 4.5 (QF Non–Universality—Purely Smooth Core))**.** *Assume the discrete core is trivial ($B = \{*\}$) and that interfaces never implement threshold events. Suppose further that the evolution is C^1, Lipschitz, K–equivariant with $K \cong (S^1)^m$, and energy–bounded so that Z is constant (no phase slips). Then the induced dynamics reduces to rotations on a compact torus $T \cong (S^1)^m$, possibly with coupling, hence is periodic or quasi–periodic. In particular, there is no simulation of an unbounded universal update scheme (no TM universality).*

Proof sketch. With B absent and no threshold events, the state space per cell is a compact abelian group T with smooth K–equivariant flow. Energy bounds freeze Z, so the global state is a product of tori with C^1 Lipschitz evolution given by abelian phase transport. Such systems are equicontinuous and (in this abelian setting) conjugate to rotations; their orbit structure is periodic/quasi–periodic and lacks the unbounded discrete memory required to simulate a universal CA/TM (e.g., no unbounded counters). □

Remark 4.8 (Boundary of the QF Band)*.* The obstruction disappears exactly when either (a) a nontrivial B–layer with a clopen partition is reinstated, enabling discrete memory, or (b) non–abelian monodromy/threshold events are allowed (exiting QF), enabling topological state changes via Z.

It is at this point necessary to issue some disclaimers regarding the use of full S^1 cyclic logic gates, that is, the $SO(2)$ corridor. Within the quasi–fluidic band, the orientation layer may be taken either as a finite cyclic alphabet C_N or as a continuous phase $SO(2)$. Passing to the full S^1 limit is mathematically tidy (abelian, compact), but it also oddly backfires by creating a regime with atypical E/V tradeoff behavior. The following remarks record the current hypothesis of this paper.

Definition 4.9 (The $SO(2)$ State Space and the E/V Paradox)**.** Fix a readout and interfaces as in the QF hypotheses. Then the following holds.

(a) **Low verifiability** — due to near resonance, small phase errors produce large misclassifications under a fixed partition; global phase–locking also removes independent redundancy ("everyone shares one clock"), degrading cross–checks.

(b) **Low expressiveness** — because commutativity collapses sequencing power: with only abelian $SO(2)$ phases, one cannot exploit noncommuting compositions; algebraic closure grows slowly, and the discrete core must carry most of the complexity.

(c) **Transition, not destination** — relegates the pure S^1 limit as a clean corridor before the non–abelian regime. Without thresholds, it reduces to periodic/quasi–periodic dynamics (4.5); with a clopen partition, it regains classical universality on B (4.4).

In summary, the full $SO(2)$ band is deliberately a "critical corridor": visually orderly, yet fundamentally brittle (low V), and shallow in purely geometric expressiveness (low E) unless discretized. It is best regarded as a staging layer for robust discrete computation on B rather than as the computation substrate itself.

4.6 The Algebra of Flow

Starting in the quasi-fluidic regime, interactions are not purely symbolic: they are geometric and dynamical—the information is encoded in the differential structure of both flow and form. It is at this point that the cross-sectional geometry, as discussed in Section 2.6, reappears.

How, then, should we define computational expressiveness in this new context? Thus far, for the sake of simplicity, we have refrained from using cross-sectional geometry as an encoding variable in the FHCM. In the future, however, it will be formalized as follows:

Cross Section Geometry $\Xi(t) \in \mathbb{R}^+$: A normalized measure of *macro-expressiveness* in the computational phase space. It quantifies the total geometric variation in a cross-section compared to an isotropic flow volume.

Definitions of expressiveness are notoriously subtle. Using harmonic analysis, we could formalize expressiveness mathematically for cross-sectional geometry itself, but now, we must operate in a setting that intermingles logic with geometry. Here, Felleisen's theorems on the expressive power of programming languages [9] once again prove decisive. To construct a geometric analogue of his definition, we consider the ability of one system of logic to engage in *macro-expression* of another.

It is a common misconception that Turing completeness "flattens" all systems of computation in terms of expressiveness. Felleisen demonstrated this is not the case: crucial distinctions exist between computational systems at the level of core operational constructs—not merely as "syntactic sugar."

As previously discussed in Section 2.6, binary logic is the most expressive form of discrete logic—less is more in this regime, and we see it in geometry. Using binary logic gates, one can construct any other logic gate, but the reverse is not generally true, especially when considering higher n-ary logics such as quaternary or beyond. Once again, logic and language are the same in geometric computation: any computational process implementable in hardware can exist in software. One need only look to the history of graphics hardware and APIs to observe this ebb and flow.

Remark 4.9. Dear reader, the author understands that this definition of Ξ may raise eyebrows, but it is *not* an error. Is it possible to calculate the exact value of macro-expressiveness? Rice's theorem says no, and this we do not dispute. In real systems, noise always intervenes, but this does not preclude a normalized measure of expressiveness itself. We contend that such a measure is analogous to the way physics treats entropy: there is no closed-form solution for entropy in general, yet we know entropy increases. Thus, we build a field (thermodynamics) around a guiding principle, not a precise formula.

Theorem 4.6 (Fluidic Construction of E/V Tradeoff)**.** *Inherent tradeoffs between geometric expressiveness and algebraic verifiability arise in the idealized computational flow of an incompressible fluid within the cross-section of a geometrically computable manifold.*

Proof. Let the cross-section be a disk D^2 in the xy-plane. The velocity field $\vec{v}(r,\theta)$ at each cross section can be written in polar coordinates as

$$\vec{v}(r,\theta) = v_r(r,\theta)\,\hat{r} + v_\theta(r,\theta)\,\hat{\theta},$$

where v_r and v_θ are the radial and angular components, respectively.

Suppose the flow is smooth and incompressible, and consider the angular dependence of v_r and v_θ. Each component admits a Fourier expansion:

$$v_r(r,\theta) = \sum_{k=-\infty}^{\infty} a_k(r)\,e^{ik\theta}, \qquad v_\theta(r,\theta) = \sum_{k=-\infty}^{\infty} b_k(r)\,e^{ik\theta}.$$

Imposing rotational symmetry of order n means requiring invariance under rotation by $2\pi/n$:

$$v_r(r,\theta + 2\pi/n) = v_r(r,\theta) \qquad \forall\,\theta.$$

This forces all Fourier coefficients $a_k(r)$ and $b_k(r)$ with $k \notin n\mathbb{Z}$ to vanish. Thus, only harmonics with k divisible by n remain:

$$v_r(r,\theta) = \sum_{m=-\infty}^{\infty} a_{nm}(r)\,e^{inm\theta}.$$

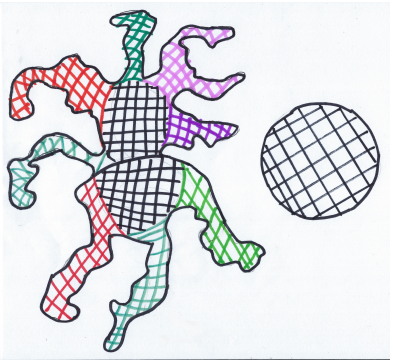


Figure 4: Macro-expressiveness vs Isotropy

In the absence of symmetry ($n = 1$), all $2K + 1$ modes with $|k| \leq K$ are permitted. Under n-fold symmetry, only those with $k = 0, \pm n, \pm 2n, \ldots, \pm Mn$, with $M = \lfloor K/n \rfloor$, survive—yielding $2M + 1$ allowed modes.

We define the *relative expressiveness* as the fraction of permitted angular degrees of freedom:

$$E(n) := \frac{\text{Number of allowed modes for symmetry } n}{\text{Number for } n = 1} = \frac{2M + 1}{2K + 1}.$$

For example, with $n = 4$ and $K = 8$, we have $M = 2$ (i.e., $k = 0, \pm 4, \pm 8$), so $E(4) = 5/17 \approx 0.29$.

As $n \to \infty$, $M \to 0$ and $E(n) \to 0$; i.e., only the constant mode survives. The only flow invariant under arbitrary rotation is uniform, and all angular structure is lost. This directly parallels the earlier result that the only smooth closed 1-manifold with infinite rotational symmetry is the circle.

Thus, increasing symmetry necessarily reduces the space of physically expressible patterns. The tradeoff is exact, quantitative, and determined by the order of symmetry imposed on the system.

□

Now, the internal geometry of the cross-section becomes a new axis of calculation. We can have different flow channels, topological defects, twists, or even branching flow lines within the same cross-section, if the shape allows it. Shapes more complicated than circles allow for distinct internal flow patterns that simply cannot exist in a circle. The more expressive cross-section geometry can allow for far more "channels" of fluid flow. Computation could be structured in more expressive ways that are simply not possible with an isotropic volume. It's not just which logical states, but how the flow moves *within* the logic state.

This is precisely the geometric analogue to Felleisen's macro-expressiveness we needed. If micro-expressiveness asks "How many states can a bit have?" then macro-expressiveness demands we answer "But how many different kinds of computation can happen, depending on the internal geometry?"

Remark 4.10. **Warning.** The E/V proof constructed via geometric-algebraic comparison of flow channels belongs fundamentally in the **fully fluidic regime**. Unlike the first two abelian regimes—where ω-logic directly encodes bit states and geometry serves only as constraints—in the fully fluidic regime, the geometric complexity of flow patterns themselves directly encodes computational logic and expressiveness. Exploring the fully fluidic regime is something we are not equipped to handle yet, but using the most general of all equations, we can glimpse at the power it has in store. Even though the use of 4-fold symmetry would not truly be considered in the fluidic mode, this construction is primarily intended to prove a specific theorem.

Comparing the extreme ends of the spectrum, there are, in fact, two fundamentally different regimes of analog computation. The first (algebraic analog flow) arises in systems governed by smooth, algebraic evolution [16], where states are continuous but transitions are predictable and algebraically controlled. The second (geometrical chaos) emerges in systems with sensitive dependence on initial conditions, nonlinear feedback, or singularities, where computation is encoded in the structure of the chaotic trajectories themselves. Only the latter regime can transcend the expressive power of algebraic machines, accessing the uncountable and unpredictable, as seen in fluidic and geometric chaos. So

[16] This is the BSS model, which will be discussed later.

not all forms of analog computation are created equal, in the same way that Cantor showed there are more real numbers than natural numbers. Pour-El and Richards pointed directly at this issue in their famous paper [20], but the spectrum of n-logics we can derive in GCT from $SO(2)$ allows a more easy visualization of what precisely they identified.

Corollary 4.7 (Twin Helix Computation Twist–Expression Boundary)**.** *For any computational twin helix, there exists a natural boundary between its geometric expressiveness and its twist density.*

Proof. Let $\mathcal{H}(t)$ be a smooth helical segment in the FHCM, and define:

Cross-sectional volume $V_{\mathbf{cross}}(t)$ — the volume of the cross-section at time t.
Embedding dimension D — the dimension of the ambient space (typically $\mathbb{R}^3$ or $\mathbb{R}^4$).

Suppose, for contradiction, that the set of all possible cross-sectional geometries includes all closed 1-manifolds. Since the twist density τ implies periodicity along the helix, at sufficiently high twist rates, there would exist (by the pigeonhole principle) two instances of an infinite set of cross-sectional geometries that must overlap or intersect within $\mathcal{H}(t)$, resulting in singularities.

Therefore, there exist maximum allowable values $\tau_{\max}$, $\Xi_{\max}(\tau)$ (cross-sectional expressiveness), and $V_{\max}(\tau)$ such that

$$\text{if } \tau(t) > \tau_{\max} \implies \Xi(t) < \Xi_{\max}(\tau(t)) \quad \text{and} \quad V_{\text{cross}}(t) < V_{\max}(\tau(t)),$$

with equality only in the limiting case of perfectly cylindrical (minimal expressiveness) and maximally twisted helices. □

Remark 4.11. One cannot simultaneously encode more twists and more geometric expressiveness within the same physical space. Increasing twist density reduces both the allowable expressiveness and cross-sectional capacity. This constraint holds even in higher-dimensional embeddings D, since each unit helix must preserve its own geometric consistency and avoid self-intersection or fluid shear instability. The total expressiveness and memory capacity of a unit helix is thus bounded from above by geometry, and cannot be made arbitrarily large at high twist rates.

Indeed, this pattern mirrors what is observed in highly dynamic programming languages: they often run more slowly than static ones, unless equipped with specialized hardware. See Kogge [21] for a technical discussion of how this phenomenon has historically played out in computer architecture. Steele and Sussman [22] provide another implementation. Future work will involve understanding this quantum boundary better to develop high-level language CPU extensions to the RISC-V architecture for a next-generation operating system.[17]

Remark 4.12 (Closing Remark on the EVT Hypothesis). Proving the expressiveness/verifiability tradeoff for finite state or non-Von Neumann models is an exercise suitable for graduate coursework. Extending the result to full Von Neumann architectures and modeling the curve between programming languages is the realm of doctoral dissertations or collaborations between professional research mathematicians and computer scientists with backgrounds in human-computer interaction. But to prove the general EVT Hypothesis at the categorical level, capturing all forms of computation, digital or analog? That would require mathematical insight worthy of a Fields Medal. So, concerning the EVT Hypothesis, here we must stop, as this topic is a vast metamathematical maze, and we wish to stick to the specific properties of twin helices for the remainder of the paper.

4.7 Application of Lie Groupoids to the Quasi-Fluidic Regime

As the state space of transformations in large abelian (C_n) geobit groups begins to resemble compact simple Lie groups in the "fuzzy" logic limit, and since Kleene's theory of partial recursion suggests groupoids as the most natural structure for partial transformations, it is natural to ask: Can these perspectives be unified? The answer is yes. The concept of a *Lie groupoid*—first introduced by Charles Ehresmann in the 1950s—provides exactly this synthesis.

A Lie groupoid is, informally, a groupoid in which the space of objects and the space of morphisms are smooth manifolds, and all structural maps (source, target, identity, inversion, and composition) are differentiable. Like Lie groups, they are built on differentiable manifolds. Still, as groupoids, they

[17] "Professor, why is the Python code running slow?" "Aha, well you see, the geometry of a Yang-Mills field is such that there are these computational helices and uh...look, this one is more complicated, okay?"

allow for a more flexible, local symmetry structure: instead of a single global group action, we have objects and invertible morphisms between them. The classification of orbits and stabilizers in a Lie groupoid encodes a richer computational structure, offering a spectrum of intermediate computational classes bridging the digital/symbolic and analog/fluidic extremes.

A definition is formulated below that serves to ground the underlying fundamentals of a GCM, a category whose morphisms are manipulations of homotopies "decorated" by the underlying geometry.

Definition 4.10 (S–decorated Groupoid)**.** Let $(S, *)$ be a monoid with unit e, and write $\mathbf{B}S$ for the one–object category whose endomorphism monoid is S with composition given by $*$. An *S–decorated groupoid* is a pair $(\mathcal{G}, d)$ consisting of a small groupoid $\mathcal{G}$ and a functor

$$d : \mathcal{G} \longrightarrow \mathbf{B}S.$$

Equivalently, each morphism $f \in \mathrm{Mor}(\mathcal{G})$ is assigned a label $d(f) \in S$ such that $d(g \circ f) = d(g) * d(f)$ and $d(\mathrm{id}_x) = e$ for all composable f, g and objects x. Since functors preserve isomorphisms, $d(f)$ lies in the group of units $S^\times \subseteq S$ for all f.

Definition 4.11 (Morphisms of S–decorated Groupoids)**.** A morphism $(\mathcal{G}, d) \to (\mathcal{H}, \delta)$ of S–decorated groupoids is a functor $F : \mathcal{G} \to \mathcal{H}$ such that $\delta \circ F = d$. This defines the category $\mathrm{DecGrpd}_S$ of S–decorated groupoids.

Remark 4.13. The use of the term "decorated" in this work is inspired by its widespread adoption in modern geometry and category theory (see, e.g., Penner [23], and Fong [24]). In particular, Baez's [25] construction of "decorated cospans" provides a categorical framework for attaching auxiliary data (the decorations) to the objects or morphisms of a base category, preserving compositionality and enabling a rich interplay between structure and process. The notion of a decorated groupoid used here is entirely in this spirit: we enrich the fundamental groupoid with additional geometric or computational data, compatibly with groupoid composition and equivalence.

When dealing with a groupoid, we should ask what its fundamental invariants are. Let $\mathrm{THM} \cong S^1_\theta \times D^2_{(r,\varphi)}$ be the solid–torus twin helix with core circle $S^1 \times \{0\}$. The projection $\mathrm{pr}_{S^1} : S^1 \times D^2 \to S^1$ induces a winding–number map on paths: for a path $\gamma : [0,1] \to \mathrm{THM}$ with endpoints $x, y \in \mathrm{THM}$, the integer:

$$\omega([\gamma]) := \deg\big(\mathrm{pr}_{S^1} \circ \gamma\big) \in \mathbb{Z}$$

depends only on the homotopy class $[\gamma] \in \Pi_1(\mathrm{THM})(x, y)$. Consequently, for any x, y the hom–set $\Pi_1(\mathrm{THM})(x, y)$ is a $\mathbb{Z}$–torsor, and each isotropy group $\Pi_1(\mathrm{THM})(x, x)$ is canonically isomorphic to $\mathbb{Z}$. In particular, the (undecorated) fundamental groupoid is determined by winding about the core circle.

To encode the helical geometry used in computation, we equip THM with orientation/chirality structure and an $SO(2)$–phase, yielding a decoration target: S (e.g. $S = \mathbb{Z} \times C_3$ or $S = S^1 \times C_3$). This gives a functor:

$$d : \Pi_1^{\mathrm{thin}}(\mathrm{THM}) \longrightarrow \mathbf{B}S, \qquad [\gamma] \longmapsto \big(\text{winding/phase along } \gamma,\ \text{chirality along } \gamma\big),$$

which is constant on thin homotopies. Thus, the homotopical data relevant for GCMs is not merely the abstract $\mathbb{Z}$–valued winding from Π_1, but the *decorated* classes $\big([\gamma],\, d([\gamma])\big)$ that package the explicitly parametric helical geometry.

In particular, if $p_A, p_B \in \mathrm{THM}$ are the distinguished apex points of the two unit helices, then $\Pi_1(\mathrm{THM})(p_A, p_B)$ is a $\mathbb{Z}$–torsor measured by net turns about the core; the decoration d refines this by recording, for each representative path, the (chirality–resolved) twist data prescribed by the helical geometry. Any deformation through thin homotopies with endpoints fixed leaves d unchanged; changes in d arise only from altering the geometric structure (e.g. switching chirality or modifying the phase law), not from the bare homotopy class alone.

Definition 4.12 (Decorated Fundamental Groupoid, Informal)**.** Let M be a smooth (or stratified) manifold, and let $\mathcal{S}$ denote a specified set of geometric, combinatorial, or dynamical structures on M—such as winding numbers, twists, chiralities, orientation data, flows, or other local parameters.

The *decorated fundamental groupoid* $\Pi_1^{\mathcal{S}}(M)$ is defined as follows:

(a) **Objects** — are points $x \in M$ (as in the classical fundamental groupoid).

(b) **Morphisms** — are homotopy classes of continuous paths $\gamma : [0,1] \to M$ from x to y, together with an assignment of "decorations" from $\mathcal{S}$ to each path, compatible with the composition law. That is, a morphism from x to y is an equivalence class $[\gamma, d]$, where d records the geometric or combinatorial data along γ.

(c) **Composition** — is given $[\gamma_1, d_1] : x \to y$ and $[\gamma_2, d_2] : y \to z$, their composite is $[\gamma_2 * \gamma_1, d_2 * d_1]$, with decorations composed via a specified rule (e.g., addition of winding numbers, concatenation of twists, etc.).

Two decorated manifolds $(M, \mathcal{S})$ and $(N, \mathcal{S}')$ have equivalent decorated groupoids if there exists a functor between their groupoids preserving both the path composition and the decoration data.

In practice, the decoration $\mathcal{S}$ can be:

- A winding number function assigning an integer to each loop (e.g., in $\pi_1(M)$ or its universal cover).
- A combinatorial label (such as twist, chirality, or a physical field value) defined locally or globally along paths.
- A program, automaton, or any computable process attached to each morphism, as in synthetic homotopy type theory.

The classical fundamental groupoid is recovered by taking all decorations to be trivial (i.e., move into a coordinate-free geometry space).

In a GCM, the fundamental groupoid is a combinatorial-topological hybrid: it records both geometric (number of twists) and logical (chirality) data.

Example 4.2 (Fundamental Groupoid and Covering Groups). Fix $S = \mathbb{Z} \times C_3$ with operation:

$$(\tau_2, \chi_2) * (\tau_1, \chi_1) \;=\; (\,\tau_1 + \tau_2,\; \chi_1\chi_2\,),$$

where $\tau \in \mathbb{Z}$ encodes net twist count (in units of 2π) and $\chi \in C_3 = \{0, \pm 1\}$ encodes chirality. For a smooth manifold M, one may refine to the thin fundamental groupoid $\Pi_1^{\text{thin}}(M)$ and define a geometric decoration:

$$d :\ \Pi_1^{\text{thin}}(M) \longrightarrow \mathbf{B}(\mathbb{Z} \times C_3), \qquad d([\gamma]) \;=\; \big(\tau(\gamma),\, \chi(\gamma)\big),$$

with path reversal sending $(\tau, \chi) \mapsto (-\tau, \chi^{-1})$. In general, d does not factor through the ordinary $\Pi_1(M)$, since $\tau(\gamma)$ can vary under thin homotopy while remaining constant under ordinary homotopy.

So, for a given path $[\gamma] \in \pi_1(M)$ in a twin helical manifold, the decoration is explicitly computable:

$$[\gamma] \mapsto (\tau_A \cdot \chi_A) + (\tau_B \cdot \chi_B)$$

where A and B index the two unit helices.

To study the ensemble or statistical behavior across a collection of such helices (or geobit types), we may define the average decorated invariant:

$$\langle [\gamma] \rangle = \frac{1}{n} \sum_{i=1}^{n} \tau_i \cdot \chi_i$$

where n is the total number of possible configurations in the groupoid space.

In the fully symmetric regime (V_4), the average winding number over all geobit types is $(0 \times 2 + 2 \times 2)/4 = 1$. In more general or dynamical regimes, where twist and chirality are imbalanced or can vary more freely (e.g., ternary undefined space in chirality eliminates winding numbers on half the manifold), this average may take arbitrary values, and the distribution of invariants can be studied using tools from measure theory and ergodic theory.

Definition 4.13 (Homotopy Computation, Informal)**.** Let $(M, \mathrm{str}) \in \mathcal{ESM}^+$ with an S–decoration $d : \Pi_1^*(M) \to \mathbf{B}S$ (or $d : G \ltimes M \to \mathbf{B}S$). A *homotopy computation* is a decorated path/state evolution $\gamma : [0,1] \to \mathcal{S}_{\text{GCM}}$ whose updates are given by the arrows of the decorated groupoid, and whose outcome is the composite label $d(\gamma) \in S$. Two computations are considered equivalent if there exists a 2–cell (thin homotopy rel. endpoints) $H\gamma \Rightarrow \gamma'$ along which d is constant, so they have the same outcome in S.

Remark 4.14. This definition is by intent not fully formalized, but it should be noted that the homotopy is such that:

- The path encodes evolution under flows/diffeomorphisms/group actions.
- Homotopies of computations correspond to deformations preserving the S–invariants.

So, even when the underlying space is topologically trivial (e.g. $THM \simeq B^3$), a choice of geometric structure induces a decoration:

$$d : \Pi_1^{\text{thin}}(M) \longrightarrow \mathbf{B}S,$$

so the S–decorated groupoid $\Pi_1^S(M, \text{str})$ is typically nontrivial. Hence, the computational content lives in the decoration d, not in the bare homotopy type of M. Composition must respect both topology and S–labels, yielding a stable, compositional invariant that records the history of transformations.

In lattice models, one naturally obtains a diagram of interacting decorated groupoids $\{\Pi_1^S(H_i, \text{str}_i)\}$ coupled by flow functors; this forms the state–transition layer of the computation. Much of the theory is faithfully captured at the 1–categorical level; higher categorical structure is added only when coherence or field–theoretic extensions (decorated bordisms, evaluation functors Z) are required.

Heuristically, it seems that transcendental geometry encodes computation, whereas algebraic topology organizes proofs. This is the practical duality guiding the framework.

Conjecture 4.8 (Extended Grothendieck Homotopy Hypothesis)**.** *Let M be a smooth or stratified manifold equipped with additional geometric or combinatorial structure (such as twist, chirality, or orientation), and let $\mathcal{G}$ denote its* decorated fundamental ∞-groupoid*: the structure encoding homotopy classes of paths together with all computational, combinatorial, or dynamical data attached to those paths. Then the following hold:*

(a) ***Classical Recovery:*** *If $\mathcal{G}$ is stripped of all computational or combinatorial decorations, it reduces (up to equivalence) to the classical ∞-groupoid of M, recovering the standard Homotopy Hypothesis: "Homotopy types are equivalent to ∞-groupoids."*

(b) ***Enrichment:*** *The enriched groupoid $\mathcal{G}$ encodes not only the homotopy type of M, but also the full combinatorial, computational, or dynamical structure of the system—so that two decorated spaces are equivalent if and only if their enriched groupoids are equivalent (as symmetric monoidal ∞-categories, or in an appropriate enriched sense).*

(c) ***Failure of Classical Correspondence:*** *There exist decorated geometric manifolds M and N such that $M \simeq N$ as topological spaces (i.e., weak homotopy equivalence holds), but their enriched fundamental groupoids $\mathcal{G}_M$ and $\mathcal{G}_N$ are not equivalent in any enriched ∞-categorical sense. More formally, there exists a category $\mathcal{ESM}^+$ of smooth or stratified manifolds with flow-preserving morphisms, torsion structure, and chirality data such that:* No functor $F : \mathcal{ESM}^+ \to \infty$-Gpd induces a homotopy equivalence that simultaneously preserves path components and the torsion-invariant structure, unless F is enriched with additional framing or coordinate data.

(d) ***Computational Duality:*** *For any such enriched groupoid, there exists a "Curry–Howard dictionary" between:*

   (a) decorated paths as proofs*, and*

   (b) decorated paths as programs*,*

   so that the groupoid serves simultaneously as a classifier of homotopy types and as a substrate for geometric computation. So that: Decorated ∞-groupoids $\simeq$ Decorated topological spaces

Remark 4.15. It would be amiss to not mention Grothendieck's Homotopy Hypothesis, which equates the world of homotopy types with that of ∞-groupoids, classifying spaces up to continuous deformation via algebraic symmetry. However, in the regime of geometric computation, the fundamental groupoid now acquires a dual role—it is both a classifier of paths and a dynamical engine for computation and flow. This suggests a possible enrichment of the hypothesis, where ∞-groupoids may encode intrinsic computational or dynamical structure. A bridge between geometry, logic, and computation that Grothendieck's original vision only partially foreshadowed.

Remark 4.16 (On the Application of the EGHH vs. GHH)*.* If d factors through $\Pi_1(M)$, then:

$$\Pi_1^S(M) \simeq \Pi_1(M) \times \mathbf{B}S$$

and the geometric realization of the nerve yields a classical homotopy 1–type ($B\Pi_1(M) \times BS$ when S is a group). In the geometric (thin) case above, d typically does not factor through $\Pi_1(M)$, and the ordinary nerve–realization forgets the smooth holonomy/torsion data. To retain it, one must work in a smooth/Lie setting, motivating an extended homotopy hypothesis relating decorated ∞–groupoids to decorated topological/smooth spaces.

Remark 4.17 (Floer Homology and the Unknown Homotopy Type). As a note for future work, Floer's original construction [26] assigns a chain complex to an infinite–dimensional configuration space, yet it remains an open problem to realize that complex as the singular (or stable) homology of an actual homotopy type. In particular, it is not known whether there exists a spectrum or even an ∞–groupoid whose homology recovers the Floer homology for a given functional and whose weak equivalence class is independent of the analytic choices involved (see [27]). This "missing homotopy type" is widely viewed as the conceptual gap separating Floer theory from classical Morse theory, where the CW-complex of critical points supplies a canonical model.

GCT approaches the problem from the opposite direction by encoding flows and torsion data symbolically, so we obtain a finite presentation of the fundamental groupoid of trajectories. Should this presentation survive stabilization, it would furnish an explicit candidate for the elusive Floer homotopy type—now rendered algorithmically accessible. While I do not claim to solve the homotopy type problem here, the construction clarifies why a fully discrete model could exist at all and identifies the algebraic structures that any such model must respect.

To provide a rigorous setting for the maximal abelian regime, we explicitly construct a Lie groupoid.

Definition 4.14 (S–decorated Lie groupoid)**.** If $\mathcal{G} \rightrightarrows M$ is a Lie groupoid and S a Lie group, an *S–decorated Lie groupoid* is a smooth functor $d : \mathcal{G} \to \mathbf{B}S$, where $\mathbf{B}S$ is the one–object Lie groupoid with endomorphism group S.

Definition 4.15 (Core Manifold for the Quasi–Fluidic Regime)**.** Let:

$$M_{\text{core}} \;:=\; C_3 \times S^1 \times \mathbb{R}^2,$$

with coordinates $(\chi, \theta, \phi, \dot{\phi})$, where $\chi \in C_3 = \{0, \pm 1\}$ is the chirality bit, $\theta \in S^1$ is the orientation phase ($SO(2)$), and $(\phi, \dot{\phi}) \in \mathbb{R}^2$ are flow variables. As a product of smooth manifolds (with C_3 discrete), M_{core} is a finite–dimensional smooth manifold.

Definition 4.16 (Core Groupoid Action)**.** Let:

$$G \;:=\; C_3 \times SO(2) \times \mathbb{R},$$

acting smoothly on M_{core} by:

$$(\varepsilon, R_\alpha, t) \cdot (\chi, \theta, \phi, \dot{\phi}) \;=\; \big(\varepsilon\chi,\ \theta + \alpha \pmod{2\pi},\ \Phi_t(\phi, \dot{\phi})\big),$$

where $R_\alpha \in SO(2)$ is rotation by angle α, $\varepsilon \in C_3$ flips chirality, and Φ_t is the time–t flow of a fixed smooth vector field X on $\mathbb{R}^2$ (e.g. $X(\phi, \dot{\phi}) = (\dot{\phi}, F(\phi, \dot{\phi}))$ for some smooth F).

Definition 4.17 (Core Lie Groupoid as an Action Groupoid)**.** The core quasi–fluidic Lie groupoid is the action groupoid:

$$\mathcal{G}_{\text{core}} \;:=\; G \ltimes M_{\text{core}} \rightrightarrows M_{\text{core}},$$

whose arrows are pairs (g, x) with $g \in G$ and $x \in M_{\text{core}}$, source $s(g, x) = x$, target $t(g, x) = g \cdot x$, and composition $(h, g \cdot x) \circ (g, x) = (hg, x)$. This is a finite–dimensional Lie groupoid; s and t are submersions and all structure maps are smooth.

Definition 4.18 (Quasi–fluidic Decoration)**.** Fix the abelian Lie group:

$$S \;:=\; S^1 \times C_3,$$

encoding continuous $SO(2)$ phase and discrete chirality. Define a smooth functor (decoration):

$$d \;:\; \mathcal{G}_{\text{core}} \longrightarrow \mathbf{B}S, \qquad d\big((\varepsilon, R_\alpha, t), x\big) \;:=\; \big(e^{i\alpha},\ \varepsilon\big).$$

Thus composition in $\mathcal{G}_{\text{core}}$ corresponds to:

$$\big(e^{i\alpha_2}, \varepsilon_2\big) \cdot \big(e^{i\alpha_1}, \varepsilon_1\big) = \big(e^{i(\alpha_1+\alpha_2)},\ \varepsilon_1\varepsilon_2\big) \in S^1 \times C_3,$$

and time–translation $t \in \mathbb{R}$ contributes no decoration in the quasi–fluidic baseline. If a clock phase is desired, one may post-compose the $\mathbb{R}$–factor with a fixed homomorphism $\kappa : \mathbb{R} \to S^1$, $t \mapsto e^{i\Omega t}$, and set $d\big((\varepsilon, R_\alpha, t), x\big) = \big(e^{i(\alpha+\Omega t)}, \varepsilon\big)$.

Remark 4.18. By using an action groupoid $G \ltimes M_{\text{core}}$, we remain in the finite–dimensional Lie groupoid setting. If one instead wants all local diffeomorphisms of M_{core}, the natural object is the *étale* groupoid of germs of local diffeomorphisms, which is infinite–dimensional; we hold off on that here.

This work presents one possible interpretation of how a homotopy-theoretic framework for computation could be formulated. I offer it in the hope that it will be considered, refined, or challenged by the broader mathematical, computer science, and physics community.

4.8 Fiber Bundles and Moduli Spaces on Chiral Manifolds

We should note that the fiber bundle structure of a single twin helix manifold is trivial: as a bundle, it is globally diffeomorphic to a solid torus,

$$\text{THM} \ \cong\ S^1_\theta \times D^2_{(r,\varphi)},$$

i.e. a circle's worth of disks with a global trivialization fixed by the helical parameters.

This apparent simplicity is misleading for computation. Triviality means there is no topological twisting (no monodromy in the D^2–fibers). Consequently, when geometric computation is introduced—via operations on chirality, orientation, and flow—the information lives not in the topological class of the bundle but in the dynamics and group actions we impose by gluing, deformations, and parameter updates. In this sense, triviality is a feature: it separates "hardware" (the geometric scaffold) from "software" (the computational process).[18]

As a motivating comparison, jet bundles on the product $S^1 \times D^2$ admit global coordinates, so derivative data and feedback laws can be managed globally. By contrast, in a nontrivial fibration (e.g. the Hopf fibration), holonomy and monodromy produce path–dependent phases and constraints that act like "quantum–style" control effects, often complicating and sometimes fundamentally constraining computation.

Example 4.3 (Phase–Accumulating Ring on a Trivial Bundle). Write $THM \cong S^1_\theta \times D^2_{(r,\varphi)}$ with chirality $\chi \in C_3$. Let the quasi–fluidic update assign a local angular rate $\Omega(r, \theta)$ and define the one–turn "gate" at radius r by:

$$U(r) \ =\ \exp\Big(i \int_0^{2\pi} \Omega(r, \theta)\, d\theta\Big) \ \in S^1.$$

Two loops with the same homotopy class (one turn around S^1) but different radii $r_1 \neq r_2$ can yield distinct outputs $U(r_1) \neq U(r_2)$ whenever Ω varies with r. Thus, the computational outcome depends on geometric control parameters, not on the topological class of the loop. In the decorated groupoid picture, this is captured by the $S^1 \times C_3$–label $d([\gamma]) = (U(r), \chi)$, which does not factor through π_1.

For readers more comfortable with moduli space language, consider the case of the Archimedean spiral, the simplest chiral manifold one can form. The moduli space of the full 4D THM is extraordinarily large (requiring, in principle, a Hilbert space due to the infinity of possible cross-sectional 2-manifold joins), so it is pragmatic to work with toy models and restricted cases.

Lemma 4.9. *Bifurcation of Moduli Spaces in Chiral 1-manifold Lemma. The set of all Archimedean Spirals may not be constructed via a continuous moduli space.*

Proof. Let $\mathcal{M}$ be the moduli space of smooth, non-degenerate Archimedean spirals in $\mathbb{R}^2$, parameterized by $r = a\theta$ for $a \neq 0$. Chirality is given by the sign of the a in the curvature formula. Suppose, for contradiction, that there exists a continuous path in $\mathcal{M}$ from a right-handed spiral ($a = a_0 > 0$) to a left-handed spiral ($a = -a_0 < 0$). Then the function $a(t)$ parameterizing the path must pass through $a = 0$ for some t, at which point the spiral degenerates into a singularity. Thus, there is no continuous

[18] This does not make calculations easy once gauge symmetries are included. It only removes topological obstructions.

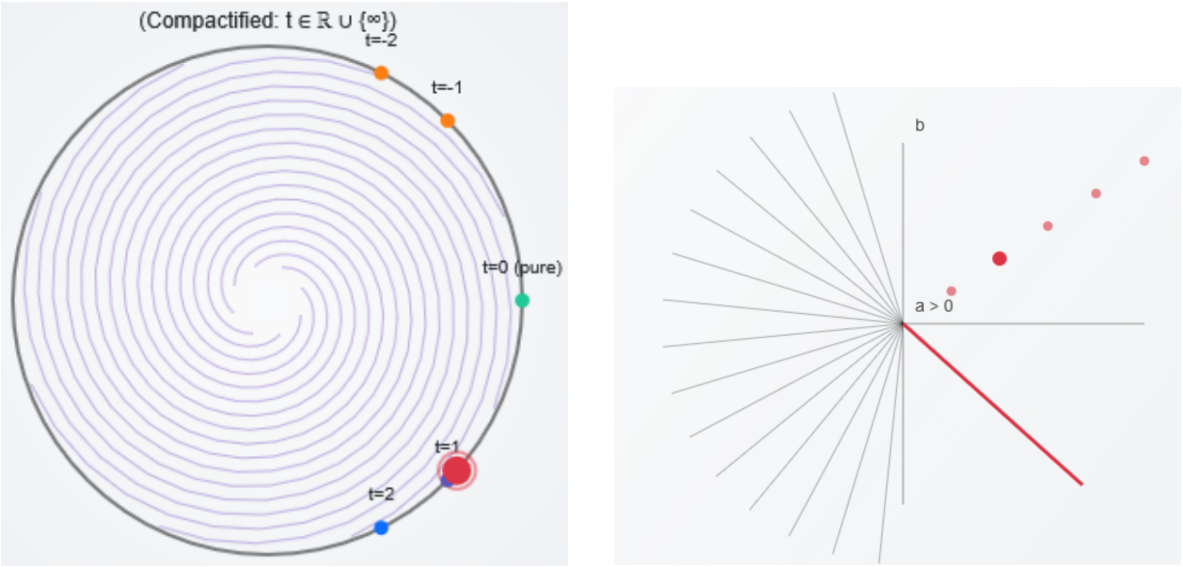


Figure 5: Visualization of the moduli space for spirals with $|a| \div |b|$. On the right, the red line encodes an instance of an adjustable ray for right-handed spirals. The red dot on the compact moduli space on the left corresponds to the phase shift. The equivalence rays opposite in grey encode possible left-handed spirals. Note the singularity at $b = 0$. The two rays cannot meet.

path in $\mathcal{M}$ connecting spirals of opposite chirality. Therefore, the moduli space splits into two disjoint components,

$$\mathcal{M}_{\text{spirals}} = \mathcal{M}_{\text{left}} \sqcup \mathcal{M}_{\text{right}}$$

corresponding to right and left-handed spirals. □

This result is related to the proof of Lemma 1.1. But now it is done in reverse. It is not possible to construct a smooth moduli space that satisfies both kinds of spirals, as doing so requires the opposite of what we previously did; rather than trying to form a continuous path, the unwinding of the geometry of the spirals causes a singularity. Though the underlying topologies are homotopic (both are smooth embeddings of S^1 in $\mathbb{R}^3$), their geometric phase structures are not deformable through a regular isotopy. But the reverse is not true. Two spirals of the same chirality can share a smooth moduli space; one only needs to account for phase shifting. The group in this case is simply C_2, the operations—multiplying a by -1 or 1. It is the barest form of a GCM.

The geometric phenomenon that splits moduli spaces by chirality, as seen in the case of Archimedean spirals, persists in much higher dimensions. Work from Mullner [28] proved the existence of strongly chiral smooth manifolds in every oriented bordism class for dimensions ≥ 3, and simply-connected examples in every dimension ≥ 7. This shows that, in general, the moduli space of such manifolds is fundamentally disconnected by chirality—there is no continuous transformation, even up to bordism, connecting a manifold to its reverse orientation.

Conjecture 4.10. ***Abelian Group-Induced Bifurcation of Geometric Moduli Spaces:*** *Let X be a smooth, compact 3 or 4-dimensional manifold constructed as the union of two helical solids joined along a common boundary, with group actions defined by independent flips of chirality and orientation on each unit helix. Let G be a finite abelian group (e.g., $\cong (C_2)^n$), representing these symmetries.*

There exists a class of such manifolds (including the THM construction) for which he space of all smooth embeddings of X into $\mathbb{R}^n$ (for $n \geq 3$) modulo G-action is bifurcated, containing at minimum two connected components not related by any smooth isotopy if not more. This bifurcation arises from G-invariant topological charges (e.g., linking parity / w_2) that persist under G-equivariant ambient isotopy; no non-abelian defects are required.

The central point underlying this is as follows: the topology of the moduli space associated with the first half of THM_4 fundamentally differs, under each group action, from the topology of the moduli

space of the second half, even though both invariants belong to the same abelian group. This structural asymmetry appears to be novel in mathematics and may offer a new perspective on the encoding of digital data in manifolds. Specifically, the ability for geometry to transition between states while the moduli space remains bifurcated by an intrinsic distinction could serve as the underlying principle for the emergence of what we call "information."

We should be mindful of this as a conjecture, given the historical difficulty of proving topological results in dimensions 3 and 4, as exemplified by the work on the h-cobordism theorem (Smale [29]). However, the highly structured, constructive nature of THM may allow for a more direct proof than is available for general manifolds in these dimensions.

4.9 Lancret's Ghost?

Helices are mathematical objects rich in dynamical variables—as explored in the FHCM—whose properties interact synergistically but without the constraints of maximal symmetry. This very lack of symmetry has led helices to be overlooked in much of pure mathematics. Yet, as we have seen, maximal algebraic symmetry inevitably destroys geometric expressiveness—the very quality needed for a dynamical system of physical computation, where symbolic encoding of state is essential. In fact, any form of computation (even total recursive proof assistants) requires at least some modicum of expressiveness, because without it, one is left only with propositional logic. Helices can dance with ones and zeros, but a sphere will never lie.

Remark 4.19. Perhaps the true reason Grothendieck and many modern geometers favor the coordinate-free approach is not merely that it eliminates chirality, but that it immunizes mathematics against Gödelian incompleteness. As long as one refrains from encoding mathematics—or, in this case, the manifolds themselves—into numbers, the door to paradox remains shut. In a world of pure objects and morphisms, there are no self-referential statements, and thus, the incompleteness theorems gain no foothold.

So, despite their ubiquity in physical systems—from DNA strands to solitons to spiral galaxies—helices are strikingly absent from modern algebraic geometry. This omission is not accidental: the standard helix is defined by transcendental functions (e.g., $\gamma(t) = (a\cos t, a\sin t, bt)$), placing it outside the algebraic category. Algebraic geometry, by its very design, privileges curves and surfaces defined by polynomials—algebraic varieties—over transcendental ones. But this prioritization systematically filters out a wide class of naturally occurring, dynamically rich shapes: those that appear constantly in physics, chemistry, and biology.

In this sense, the helix is a kind of orphan in the modern mathematical landscape: too smooth for combinatorics, too transcendental for algebraic geometry, and too infinite for finite computation. Yet, for a theory of geometric computation—especially one grounded in constructible physical objects—the helix rightfully takes center stage.

Remark 4.20. Admittedly, helices can be derided as "overly constructed" or "engineered" objects. They are most naturally described parametrically, as done earlier in this paper, or less commonly via implicit Cartesian equations. While these are closed-form descriptions, they may lack the classical elegance of curves defined purely by geometric or variational principles. The fact that helices are so "crafted" may not be accidental; it reflects that both helices and the formalism of geometric computation are fundamentally "user-centric"—whether one is specifying the parameters of the shape itself, or the quantization of the computational model.

But why helices specifically? At present, we can only speculate. There is no general form for moduli spaces of helical manifolds that exposes the deeper structure of these objects. Lancret's Theorem (1802) tells us that a space curve in $\mathbb{R}^3$ is a circular helix if and only if the ratio of its torsion τ to its curvature κ is constant along the curve, i.e., $\tau/\kappa = C$. **This centuries-old result may hide much deeper implications.** In my view, constructing and characterizing the moduli space of helical manifolds is the most urgent and challenging direction for future research. Such a moduli space is likely to be highly complex, infinite-dimensional orbifold with wild local and global structure. The expressive power of helices is remarkable: they admit a countably infinite variety of cross sections, as well as parameters for twisting, pitch, and chirality, producing a vast configuration space. Mapping this moduli space in detail will be daunting, even for a future (or present) Kontsevich. Nevertheless, understanding its structure is crucial for advancing both the theory and applications of geometric computation.

Helices are, in a sense, the maximally symmetric nontrivial curves in $\mathbb{R}^3$, interpolating between the absolute rigidity of straight lines and the perfect closure of circles. They arise from the interplay of sine and cosine, functions grounded in the geometry of the triangle, the simplest polygon that can exist without curvature. There is a paradox here: the curve that best encodes three-dimensional symmetry is ultimately constructed from the absence of curvature, via periodicity.

In physics, the triangle's significance appears in unexpected places. Certain subatomic particles, such as "glueballs" (hypothetical bound states of three gluons), suggest that the building blocks of reality assemble in triplets—a resonance with the "magic number" of geometry, or perhaps simply a reminder that nature delights in both symmetry and surprise. (For further discussion, see Section 5.6.)

While helices offer remarkable efficiency for encoding computational structure, it is important to note that not all helices in nature are used for computation, nor do they dominate large-scale architectures. Helical forms excel at packing information, facilitating transmission, and exploiting symmetry at small scales (think, DNA and chiral molecules). However, at larger scales, nature often favors other symmetries—branching, fractal, planar, or modular structures—driven by stability, energetic, and functional constraints. Thus, the computational potential of helices arises not from their ubiquity but from their unique ability to localize and propagate information when conditions permit.

Remark 4.21 (Literature Review on Helices and Research Directions). Though we have refrained from invoking specific biological or physical examples in depth here, understanding where and why helices give way to other forms remains a promising direction for research in this area and texts from computational chemistry and biology should be sought out. We recommend readers see Chouaieb et. al. [30] for a background on the role of helices in nature and their emergence. Specific attention should be paid to figures 1 and 3—a telling sign of what this moduli space might look like. Likewise, investigations should be done into the niche body of research from Japanese geometers on Killing vector fields and their helical behavior. See Adachi [31] as a leading example. As is so often the case in mathematics, the key to modern quantum theory may be quietly lurking in the diagrams of a century-old German treatise (see Loria [32]) on transcendental curves—waiting for someone to notice that helices and hyperboloids speak a universal geometric tongue.

4.10 Spooky Action, Up Close

What would a GL be able to compute reasonably better than a classic computer? Engineering problems in designing such structures at scale aside, consideration yields a rather remarkable conclusion.

Proposition 4.11. *A GL can perform quantum-like computations without entanglement or superposition, in a fully deterministic manner.*

Outline/Discussion. I understand this is a bold statement, and I must clarify at the outset that quantum-style speedups for problems requiring non-local correlations (such as factoring or unstructured search—e.g., Shor's algorithm) remain out of reach for any deterministic system. GCMs do not, and cannot, exploit quantum entanglement or wavefunction superposition.

Nevertheless, a GL can achieve "quantum-like" expressiveness by encoding highly correlated, non-separable state information directly into its geometric and algebraic structure. Each GCM possesses an internal state space determined by constraints of symmetry (chirality, orientation, topological flow), and when composed into a GL, these constraints propagate deterministically through the system. The resulting structure can simulate certain classes of computations—particularly those governed by embedded symmetry or continuous deformation—more efficiently than traditional classical digital architectures.

Unlike quantum systems, which achieve correlations via entanglement, GCMs leverage deterministic correlations: the internal degrees of freedom are coupled by group or groupoid structure. This is best understood as deterministic interference, rather than quantum interference. The group and groupoid transitions in a GL act as compact logic gates, capable of manipulating multiple bits of information in parallel, rather than the strictly local transitions of standard Boolean logic.

As the system approaches the fully fluidic regime, the computation becomes more analog: geometry and topology drive the computation via continuous, topologically constrained flows. This mode is not universal in the quantum sense, but can offer dramatic computational advantages for certain classes of problems where the problem structure matches the symmetry and constraints of the underlying geometry. □

Regarding practical speedups, GCMs may offer computational advantages in the following broad classes of problems:

Symmetry-Constrained Problems — are problems in which global or local symmetries sharply reduce the effective search space. Examples include group isomorphism, Cayley graph generation, and orbit enumeration in algebraic combinatorics. Recent advances (see Babai [33]) suggest that exploiting symmetry in problem structure can lead to dramatic algorithmic improvements—well aligned with the built-in groupoid symmetries of GCMs.
Geometric and Topological Searches — encompasses tasks where the core challenge is pathfinding or matching within highly constrained geometric or topological spaces. Applications range from motion planning in robotics (see Sola [34]) to exploring configuration spaces in molecular chemistry, as well as embedding, packing, and shape optimization problems with nontrivial smooth boundary constraints. GCMs provide a natural substrate for encoding and traversing such constrained search spaces, especially when the constraints can be directly embedded as geometric or topological invariants.
Constraint Satisfaction Problems (CSPs) with Geometric Content — can become dramatically more challenging—or tractable—depending on the geometric structure of their constraints. Classic examples include folding problems, linkage configurations, and geometric realizability of graphs (see Streinu [35]). GCMs allow the direct modeling of "fluid-like" consistency propagation, mirroring the analog solution methods that have long been powerful in geometry and engineering.

Remark 4.22. **Warning**. The final item in the above list will naturally bring up questions the author is not fully prepared to answer as of yet. While the Navier–Stokes existence/smoothness problem is not directly addressed in this paper, the theory of geometric computability provides a conceptual explanation *for why* its notorious difficulty exists. The solution space of NS is so highly macro-expressive—it is rich in geometric and computational possibilities—that it outstrips verifiability. The onset of turbulence and the possible breakdown of smoothness are manifestations of the EVT: when computation becomes too expressive, predictability collapses. Thus, it is not merely a technical challenge, but a symptom of a fundamental limit in the relationship between geometric macro-expressiveness and mathematical verifiability.

As a consequence, the act of solving it is potentially isomorphic to evaluating a subset of the infamous Busy Beaver function, embedding all the undecidability and unpredictability of Turing-complete computation. In this light, the problem is not just hard—it is the mathematical analogue of an untyped, unverifiable programming language.

Example 4.4 (Cayley Graph Generation Under Geometric Computation). As a practical illustration of geometric computational speedup, consider Cayley graph generation, motivated by Babai's algorithm as cited above, but primarily based on his earlier foundational work with Luks [36], dealing with how efficient enumeration and transformation of group structures is crucial to complexity-theoretic classification. While his construction focuses on abstract algorithmic bounds, the geometric regime allows these traversals to be realized directly in physical topology via global flow operations. This example explicitly situates itself within the quasi-fluidic regime of the FHCM model, where discrete orientation states encode group elements and global operations directly correspond to group generators.

Definition 4.19 (Cayley Graph). Let G be a finite group and $S \subseteq G$ a generator set. The Cayley graph $\Gamma(G, S)$ is a directed graph defined as follows, where *nodes* are elements of G. For each *edge* $g \in G$ and $s \in S$, there exists a directed edge from g to gs. Thus, a Cayley graph explicitly encodes the algebraic structure of the group in combinatorial form.

In classical computation, the Cayley graph is constructed by explicitly enumerating all elements and their associated edges:

Iterate over every group element $g \in G$.
For each g, iterate over each generator $s \in S$, creating an edge $g \to gs$.

This classical procedure has complexity: $O(|G| \cdot |S|)$
Classical Example: For the group $(C_2)^n$:

Nodes: 2^n
Generators: n
Complexity: $O(n \cdot 2^n)$

In the quasi-fluidic regime of FHCM, each unit (geobit) is identified with one group element. The full lattice of n-geobits directly encodes the entire group $(C_2)^n$:

Nodes: Each of the 2^n lattice sites directly encodes one element of the group.
Generators: Each generator $s \in S$ corresponds explicitly to a global geometric operation T_s applied to the lattice.

Formally, let X denote the lattice, and let the operation corresponding to generator s be denoted as:

$$T_s : X \to X, \quad T_s(g) = gs \quad \forall g \in G$$

Because each T_s acts globally and in parallel, it simultaneously creates all edges corresponding to generator s. The operation is thus a parallel group action, whose complexity is $O(1)$ per generator operation. Hence, the geometric algorithm explicitly leverages global symmetry operations rather than local enumeration.

The new geometric complexity analysis:

Apply each generator operation T_s once globally: The complexity per generator: $O(1)$. The total complexity for all generators: $O(|S|)$
Readout or explicit enumeration of all edges: still requires $O(|G|)$, but the construction itself is $O(|S|)$.

Therefore, the geometric complexity is dramatically reduced to $O(|S|)$.

Table 9: Classical vs. Geometric Computation in $(C_2)^4$

Classical Computation	**Geometric Computation**
Nodes: 16 (all group elements)	**Nodes:** 16 (all group elements, initialized in parallel)
Generators: 4	**Generators:** 4
Edges: $16 \cdot 4 = 64$ (each generator applied to each node individually)	**Global Operations:** Each generator is applied once globally, generating all edges simultaneously
Total Operations: $O(64)$ (one per edge)	**Total Operations:** $O(4)$ (one per generator)
Complexity: $O(nk)$ for n nodes, k generators	**Complexity:** $O(k)$ (global, parallel application of each generator)

Using our imaginary programming language:

```
def fun cayley-populate-row(group, row) {
    let: generators ⊂ Aut(group)
    let: lattice → Grid[GCM]
    let: state-init → group.identity()

    // Initialize each column in the row with a unique group element
    Foreach col in lattice.columns {
        let: g ← group.enumerate(col)
        set: lattice[col, row] := GCM[g, state-init]
    }

    // Apply each generator to the row in parallel
    Foreach gen in generators {
        broadcast: lattice[row] {
            act: GCM ← GCM.apply(gen)
        }
    }

    return lattice[row]
}
```

So to summarize, the Cayley graph of the abelian 2-group $(C_2)^n$ can be generated within an n-geobit GL by applying exactly one global symmetry operation per generator. The complexity is thus: $O(n)$. This yields an exponential speedup compared to the classical enumeration, which is: $O(n \cdot 2^n)$

Conjecture 4.12 (Geometric Speedup Conjecture for Abelian Cayley Graphs and Geometric Computation Complexity Time)**.** *Within the framework of GCT, certain subclasses of the Graph Isomorphism problem—particularly those arising from abelian 2-groups and their Cayley graphs—may admit an effective geometric encoding that bypasses classical enumeration. While this does not imply that GI is solvable in polynomial time in the standard Turing model, it suggests that under geometric computation, these instances can be traversed and constructed with asymptotically fewer physical operations, potentially indicating a distinct computational regime not captured by standard complexity classes.*

In light of the existence of GCMs, we should not view quantum-like computation as a purely quantum phenomenon; rather, it illustrates that the encoding of bit states as points on a manifold with multi-axial symmetry can arise in purely classical, deterministic geometric systems. This suggests that the abstract structure of qubit computation is a special case of a more general geometric logic, rather than an exclusive hallmark of quantum theory. GCMs show that the essential "logic" of a qubit is just the logic of a system whose state space is a sphere (or, in our case related manifold) with a nontrivial symmetry group. I understand that is a bold claim, and again, I do not mean to diminish the speedups granted by entanglement, as quantum computing still outstrips the power of classic or geometric computing, but it is possibly more of an ultra-powerful edge case at the end of a continuum. See Tang [37] for a case study that demonstrates why clever deterministic approaches can perform on par with quantum computers.

Remark 4.23 (Programming Language Outlook). Developing a programming language suited to actual geometric computation remains a substantial challenge. However, physical realizations of non-Von Neumann computing are not without precedent. While these machines built by MIT hackers of yore may be mostly forgotten, the "tagged token" dataflow computer architectures of the 1980s [38] are an instructive example of deterministic, non-standard models. Comparable experimental attitudes informed projects like Butterfly LISP, a dialect engineered for parallelism across heterogeneous architectures such as the BBN Butterfly and Symbolics LISP Machine [39]—demonstrating[19] explicit parallelism in software, even if the hardware was only partially faithful to the ideal.

For an actual language, we propose that a dialect such as "GeoLisp"—a LISP variant with Erlang-like concurrency and native support for group and groupoid operations—would be a natural candidate. LISP has already demonstrated its adaptability to non-Von Neumann paradigms, including quantum computing [41].

While large-scale geometric computers may depend on future advances in nanotechnology or materials science, it is hoped that the framework presented here may inspire new algorithms or pedagogical approaches. Geometric computation is richer than most standard approaches to non-VN computation, like cellular automata, while remaining deterministic, which avoids much of the confusion that comes with quantum computing. So even if practical realizations remain distant, these ideas may help to teach non-VN computation to students in an intuitive way and bridge group theory, topology, dynamical systems, and category theory.[20]

4.11 Chaos Unleashed

After assembling THM in the abelian, quasi–fluidic band, where computation is well captured by finite local operations on geometric primitives, we now view the helix lattice as a bundle with connection. Concretely, let:

$$\pi : P \to B$$

be a principal G–bundle over a computably generated base B (the working lattice region), with associated state bundle $E := P \times_G \mathsf{F}$ whose fiber F encodes the finite–dimensional internal variables of a twin–helix cell. A connection A on P (locally a $\mathfrak{g}$–valued 1–form) prescribes parallel transport of states; the curvature: $F = dA + A \wedge A$ measures the defect of path-independence. In this language, a "join" is no longer a purely combinatorial operation: it is the composition of parallel transports along

[19] Note for the DARPA/CS crowd: understanding the historical "hardware intolerance" of highly expressive programming languages like LISP towards commodity hardware was part of the motivation for this research. This was noted as far back as 1966(!) by Peter Landin in his infamous paper [40]. It turns out the reason (i.e., the EVT Hypothesis) is, as we can see, complicated. Previous comments on RISC-V extensions for Neo-Lisp Machines were not in jest.

[20] The author would like to note that the process of writing this paper has clarified some aspects of non-Von Neumann computation for himself as well, even if just a bit. And frankly, if this paper is right, we ought to teach computation the same way the universe behaves. Geometry first, quantum second.

adjacent interfaces. When G is abelian (e.g. $U(1)$ or $SO(2)$), joins commute; once G is non–abelian, joins become order–sensitive, and composition acquires memory.

Example 4.5 (Holonomy as Computational Memory.). For a loop γ based at $x \in B$, the parallel transport:

$$U_\gamma \;=\; \mathcal{P}\exp\Big(\int_\gamma A\Big) \in G$$

is the *holonomy* of the connection. The subgroup: $\mathrm{Hol}_x(A) \subseteq G$ generated by all such U_γ is the effective symmetry seen by the computation at x. In the abelian limit, $U_{\gamma_1}U_{\gamma_2} = U_{\gamma_2}U_{\gamma_1}$ and only the (additive) flux matters; in the non–abelian case, $\mathcal{P}$ (path ordering) is essential and different execution paths encode different group elements, even for homotopic loops. Thus:

(a) **Registers as loops** — are minimal cycles in the lattice (around/through a twin–helix cell) carry holonomies $\{U_{\gamma_k}\}$ that function as state registers.

(b) **Updates as transport** — are driving fields change A and hence update $\{U_{\gamma_k}\}$ via parallel transport; gluing cells modifies the loop basis.

(c) **Curvature as storage** — F stores "what cannot be gauged away," i.e. the persistent, path-dependent portion of state, that is, the geometric "memory."

This is the geometric analogue of Wilson loops in gauge theory: the observable is the holonomy, not the potential itself. In the quasi–fluidic abelian band, holonomy reduces to a scalar phase (e.g. $SO(2)$ orientation logic and integer twist counters). Entering the fluidic/gauge band promotes these phases to matrix phases (non–abelian Wilczek–Zee–type transport), and join symmetry is effectively broken by $[U_{\gamma_1}, U_{\gamma_2}] \neq e$.

Computation now proceeds by progressively internalizing curvature; symbolic joins become ordered transports, and state is read off from conjugacy classes of holonomies. The expressive resources exceed those of purely symbolic models in the sense that time–ordered composition, loop creation/annihilation, and curvature redistribution yield behaviours not well captured by classical finite–step semantics.[21] Physically, one may regard the twin helix fiber as the matter degree of freedom and A as the control field; non-commutativity (and hence path dependence) is the resource that turns join algebra into genuine gauge computation. In summary:

(a) **Symbolic/Quasi-Fluidic regime** — G effectively $U(1)/SO(2)$, joins commute, memory is a scalar phase or integer flux.

(b) **Fluidic/Gauge regime** — G is non–abelian (via the helix's symmetry group and interface groupoid), joins do not commute, memory lives in holonomy classes; curvature encodes stored computational context.

Remark 4.24. In the language of Lemma 2.4, this is when $\mathcal{E}(G)$ subsumes the majority of $\mathcal{C}(G)$. Via the chart of n-logics 7, we flip from a quasi-fluidic regime dominated by maximal abelianism to a purely fluidic computational regime dominated by non-abelian structure. The shift from abelian to non-abelian is when the true power of geometric computability is realized. Turing completeness is surpassed again at this point. The actual shape of the geometry starts to resemble an increasingly unstructured flow of water. Fully chaotic GCMs start to resemble streams of endless information.

Going further, consider the case where a GCM loses all forms of quantization. As discussed earlier in 2.6, there is a degenerate case corresponding to a GL in a 1×1 lattice. If a single GCM could lose all notion of quantization, its information-theoretic capacity would become unbounded. Such an object would, in some sense, become a GL—possessing arbitrary geobit capacity greater than any discrete GCM, but without the use of symbolic composition with any other GCM.

However, if we assume infinite capacity in a single GCM, a fundamental limitation emerges: the ability to read out or write information vanishes. Computation subsumes communication—computation may be infinite, but communication becomes impossible without a readout interface.

Conjecture 4.13 (Computation–Communication Tradeoff in GCT). *Let $\mathcal{G}(J)$ be a geometric computation model (GCM) with quantization scale J (larger J = finer resolution, continuum limit $J \to \infty$). Define:*

[21] We deliberately refrain from machine–model claims here: the point is expressivity and programmability via $A, F, \mathrm{Hol}_x(A)$ rather than any specific "completeness" notion.

(a) $C_{\text{int}}(J)$: *internal computational capacity (e.g., metric entropy /* log *of the number of effectively distinguishable internal states per fixed resource budget).*

(b) $C_{\text{ext}}(J; N, B)$: *maximum external mutual information rate achievable through an interface of* N *boundary connections under a fixed bandwidth/energy budget* B.

Assuming the interface resources (N, B) *are held fixed while the interior is refined, then*

$$C_{\text{int}}(J) \xrightarrow[J\to\infty]{} \infty, \qquad C_{\text{ext}}(J; N, B) = O_{N,B}(1), \qquad \frac{C_{\text{ext}}(J; N, B)}{C_{\text{int}}(J)} \xrightarrow[J\to\infty]{} 0.$$

This is the GCT analogue of limits familiar from measurement theory and control of high-dimensional chaotic systems: bounded interfaces cannot extract finite fractions of information from unbounded interiors. A fully continuous GCM is, in principle, an oracle with infinite internal description, but with a vanishing externally accessible fraction at fixed N, B. Re-introducing quantization (finite J) creates stable, addressable modes and restores practical read/write/communicate operations.

Definition 4.20 (The Join Commutator)**.** Given two admissible joins J_1, J_2 (gluing operations along interfaces), define:

$$[J_1, J_2] \;:=\; J_1 J_2 J_1^{-1} J_2^{-1}.$$

We say joins *commute* if $[J_1, J_2] = e$.

Definition 4.21 (Asymmetry of Regimes)**.** Recall again the chart of n-logics. The four regimes split into "inner" (Symbolic, Fluidic) and "outer" (Quasi–fluidic, Gauge) types as detected by the join commutator:

(a) **Quasi–fluidic (outer)** — $[J_1, J_2] = e$ and the local symmetry on phases is abelian. A K–equivariant change of variables puts interfaces in block form, so *independence is a coordinate choice* via decouple/register–normal–form.

(b) **Symbolic (inner)** — $[J_1, J_2] = e$ by construction; helices are *intentionally glued* through a shared discrete update table.

(c) **Fluidic (inner)** — generically $[J_1, J_2] \neq e$ couplings are path–dependent via non–abelian holonomy and helices are *dynamically glued.*

(d) **Gauge (outer)** — typically $[J_1, J_2] \neq e$, but there is gauge redundancy; after gauge–fixing, field lines behave as independent modules at the representation level, so *independence is a gauge choice* as the dynamics remains non–commutative.

Remark 4.25 (Gluing and the Sectors of ω–logic). There is a curious "outer vs. inner" asymmetry in the chart of n–logics. In the symbolic and fluidic bands helices are defined as glued; in the quasi–fluidic and gauge bands this need not always be the case. Under a C_2^4 embedding inside QF, joins commute yet are not always necessary (decoupling by normal form), whereas in the gauge band joins play a different role altogether (gauge transformations). Thus, the observed breakdown of compulsory gluing toward the extreme ends of ω–logic signifies that gluing is not primitive there: it is coordinate– or gauge–dependent structure, rather than a topological invariant of the computation.

In this now advanced setting, a constructive classification of computation inside TFTs is not a luxury but a necessity. The Cobordism Hypothesis tells us that fully extended TFTs are determined by fully dualizable objects in a symmetric monoidal(∞,n)-category[11]. This is an existence-and-uniqueness statement. For computation, we also need operational semantics: a functorial procedure that takes local update rules/flow data and compiles them into global invariants (phases, charges, holonomies) that are stable under the causality/communication constraints above. Concretely, decorated groupoids provide the state-and-transition layer; decorated bordisms provide the control-flow; and a symmetric monoidal functor to a computational topos (see 5.4) evaluates programs as topological processes. Dualisability supplies start/stop (coevaluation/evaluation), while the monoidal structure encodes parallel composition—precisely the ingredients a constructive theory of topological computation requires.

With that lens, the "regimes" below are not just stylistic choices; they are computational phases indexed by symmetry and transport data. As we enrich the allowed decorations and symmetry (from finite abelian labels to non-abelian and Lie transport), we climb from finite-group TQFTs (Dijkgraaf–Witten–type semantics for symbolic programs), through discrete higher-form gauge models

(quasi-fluidic computation with conserved fluxes), into Yang–Mills–like continuous fields (fully fluidic, non-abelian control of flows), and finally to simple-Lie, bundle-valued data where fully extended TFTs give the natural target semantics. Table 10 summarizes this hierarchy in GCT as each row specifies:

(a) The algebraic data that decorate paths and bordisms.

(b) The corresponding field-theoretic semantics.

(c) The geometric embedding that enforces locality and the global clock bound.

Disclaimer: this is still based only on modeling observations as of now. Future work will be needed to quantify more rigorous boundaries of relationships.

Table 10: The Full View of GCT and Topological Computation.

Group Type	**GCT Model**	**Topological Field**	**Example Groups**	**Embedding Space**
Abelian	Symbolic Mode	Finite group TQFT	$V_4, (C_2)^3$	$SO(3), SO(4)$
Maximally abelian	Quasi-Fluidic Mode	Discrete gauge theory	$(C_2)^4$	$SO(5)$
Non-abelian	Fully Fluidic, Chaotic	Continuous gauge field theory (Yang–Mills–like)	$D_4 \ltimes (C_2)^4$	$SO(6)+$
Simple Lie	Emergent Gauge Models	Fully-extended TQFT (Yang–Mills, Chern–Simons)	Manifold bundles	$SU(2), SO(N)$

4.12 Higher Dimensional Algebra and Geometric Computational Field Theories

To our knowledge, no single algebraic framework presently captures computation in GCMs across the full symbolic–to–fluidic range. By this we mean a framework that simultaneously (a) composes systems under joins—both commuting and non-commuting, (b) handles local–to–global propagation through interfaces, and (c) couples continuous transport (flows/phases) with discrete readout. Lie groupoids (and stacky variants) currently provide the most serviceable language for local symmetry, defects, and gluing, but a general categorical semantics that bridges a GCM's geometry with its computability is still lacking.

As one passes from discrete to quasi-fluidic regimes, finite algebraic invariants rapidly lose discriminative power: abelian phase freedoms and continuous deformations overwhelm purely combinatorial structure. This suggests a hybrid treatment in which dynamical-systems methods supply the evolution laws while higher-algebraic machinery supplies the compositional spine. Accordingly, we use "algebra" here in a programmatic sense: the aim is a language that unifies discrete composition, continuous flow, and topological joining within a single, compositional setting.

Remark 4.26. A viable semantics for GCMs must be simultaneously symbolic and geometric: it must (a) compose discrete states and interfaces (information–theoretic layer), (b) transport phases and flows continuously (geometric layer), (c) express gluing along boundaries and track joins, and (d) specify how update laws transform the underlying geometry (chirality/orientation/topological data).

The key obstacle is nonlocal constraint propagation across helices: interface compatibility is not decided on a single patch, and path–dependence (holonomy) persists under coarse discretization. Global states are therefore not simple products of local states but solutions of descent/compatibility conditions along the interface groupoid (as in 4.2). Consequently, naive local linearization or discretization erases the very invariants that govern computation.

What is needed? A workable mathematical language for GCMs must support four capabilities and tie each to a concrete compositional device.

(a) **Continuous transformation.** Represent smooth deformations (tube radius, phase, twist) and their flows.

(b) **Local-to-global behavior.** Track how interface–local updates assemble to global states and constraints.

(c) **Structural invariance and laws.** Express conservation and controlled change of chirality, orientation, and topological charges.

(d) **Rich morphism structure.** Compose fragments along boundaries; allow deformations as higher morphisms.

Definition 4.22 (Minimal Compositional Spine for GCMs)**.** Let $\mathbb{G}$ be a symmetric monoidal double category of open GCMs:

- **Objects** (0–cells) — interfaces/boundaries carrying readout structure.
- **Horizontal** (1–cells) — GCM fragments with specified interfaces (gluable bordisms equipped with interface rules $\mathcal{I}$).
- **2–cells** — K–equivariant deformations/flows between fragments (homotopies, pulses).
- **Monoidal product** — disjoint union $\otimes$ (parallel composition); *horizontal composition*: gluing along interfaces.

Decorate $\mathbb{G}$ by a stack $\mathcal{S}$ of state groupoids over $\mathcal{ESM}^+$ that assigns to each open set the fiber $\Sigma = B \times T \times Z$ and enforces descent along the interface groupoid. Dynamics consists of (a) *pulses* as endo–2–cells (discrete steps), and (b) *flows* as internal vector fields on $\mathcal{S}$ (continuous evolution).

The above definition provides an outline on minimal workable semantics. In addition, the following areas of mathematics come to mind as to what would help and how they would integrate:

Stacks and Descent Theory — can model local–to–global state by a stack $\mathcal{S}$ of groupoids (or higher groupoids) on the site $\mathcal{ESM}^+$: to each open U assign the state groupoid $\mathcal{S}(U)$ with K–equivariant structure (Def. 4.2); joins are encoded by descent along the interface groupoid. Nontrivial holonomy/monodromy appears as nonvanishing obstruction classes in (nonabelian) Čech hypercohomology; in QF these reduce to abelian local systems (torus phases T and integer charges Z), while in the fluidic/gauge regimes they become genuinely nonabelian. Thus global configurations are *glued objects of* $\mathcal{S}$, not cartesian products of local states.

Stratified, Topological Field Theories — associate algebraic data to topological spaces (see Lurie [42]). One can use a stratified, extended bordism category with defects (interfaces as 0–strata, GCM fragments as 1–strata, deformations/flows as higher strata) and a symmetric monoidal functor:

$$\mathcal{Z} : \mathrm{Bord}_{d,\partial}^{\mathrm{strat}} \longrightarrow \mathcal{C}$$

into a target higher category $\mathcal{C}$ (e.g. stacks, dg–categories). Functoriality under gluing expresses a "conservation of computational information" principle; observables can be organized by factorization algebras on $\mathcal{ESM}^+$. In the QF abelian subtheory, $\mathcal{Z}$ restricts to torus–valued local systems; allowing nonabelian holonomy recovers the fluidic/gauge behavior where $[J_1, J_2] \neq e$.

Synthetic Differential Geometry and Jets Bundles — would encode update laws and finite–order control by jet comonads J^k on fields (sections of bundles) and the Cartan/Spencer formalism; integrability and phase–slip events correspond to Spencer cohomology and failure of prolongation. Working internally to a (differential) cohesive ∞–topos provides infinitesimals and differential modalities (e.g. $\int$, $\flat$, $\sharp$) so that vector fields, flows, and K–equivariant transport are first–class objects. QF corresponds to abelian transport on $T \times Z$ with thresholded readout on B; fluidic/gauge add nonabelian connections and curvature.

Fluidic Type Theory *(Sketch)* — is a dependent type theory internal to the (differential) cohesive setting with:

- *Types* as stacks of decorated manifolds (helical/stratified with K–equivariant state fibers Σ).
- *Terms* as global sections (configured geometries).
- *Id-types* as path objects (homotopies/deformations between configurations).
- Π– and Σ–types as homotopy right/left Kan extensions along fibrations modeling parameterized families and gluing.
- a sequent calculus whose cut/elimination rules match horizontal/monoidal composition in the double category of open GCMs.

A semantics functor interprets programs as morphisms in the compositional spine (Def. 4.22), with readout functors to cellular automata (QF, Thm. 4.4) or torus rotations (smooth core, Prop. 4.5).

Remark 4.27. The type theory proposed here should be confused with *Geometric Type Theory*. Proposals for such a theory [22] are related to geometry of the coordinate-free variety, of which we cannot practice. Naming has been chosen to avoid collisions. For an example, see [43].

There are, of course, many avenues for future exploration. While the core formalism for basic (abelian) GCMs does not yet require the full machinery of **higher category theory**[23], its importance will rapidly grow as we move to regimes where geometric computation interpolates between multiple mathematical domains. In particular, the emergence of "geometric meta-programming"—that is, computations that act on and modify the very flows and programs underlying the GCM—makes higher categorical structure indispensable. In these fluidic and gauge regimes, one must deal with the issues of self-interactions. In the same way a quantum field interacts with itself, so can a highly expressive programming language. Here, higher morphisms provide a natural language for macros, feedback, and transformations of transformations, echoing both the flexibility of modern programming languages and the richness of gauge theory. Likewise, a **Donaldson-style cohomological approach** [44] also comes to mind as necessary, but for the time, this list must suffice.[24]

Remark 4.28. A full algebraic description of the true behavior of a GCM likely demands a synthesis beyond the reach of any single mathematician—a genuinely collaborative effort. The spirit of such a project would require both the experimental, combinatorial perspective of Wolfram and an energy we can only describe as "deeply Grothendieckian." There is, perhaps, a certain irony in this. Were Grothendieck alive today, he might feel profoundly ambivalent about this direction—even though geometric computability theory owes so much to his vision of mathematics as a seascape of ever-expanding generality.

Groethendieck famously spoke of a rising sea in which difficult problems could be drowned. However, when nature tries to frustrate humanity with phenomena like quantum mechanics, sometimes, no ocean is deep enough. To counter this, we may have to carve a new path; one using a river of flow.

[22] Nlab states—"At present it is not clear exactly how such a type theory should be defined."

[23] I am indebted to Emily Riehl for her excellent video series/lectures on the subject.

[24] Please note that I withhold examination of the work of Prof. Donaldson here not out of lack of importance, but because the formation of a "Computational Donaldson Theory" requires an *entire paper unto itself* to do his great work justice, in addition to a computational analogue of Floer.

5 Physical Applications in Geometric Computability

We now turn to the higher-level philosophy of GCMs and their implications, beginning with computational conjectures isomorphic to those known from classical theoretical computer science.

Conjecture 5.1 (Geometric Computability Thesis — Symbolic Regime)**.** *Let GL be a symbolic GCM whose local state set S is finite, whose neighborhood N is finite, and whose synchronous update $f : S^N \to S$ is computable. Then there exist computable enc/dec and a Turing machine T such that for all inputs x,*

$$\mathrm{dec}\big(\mathrm{boundary\_trace}_{\mathsf{GL}}(x)\big) \;=\; \mathrm{out}_T\big(\mathrm{enc}(x)\big).$$

Conversely, every Turing machine T is simulated by some such GL with at most polynomial overhead in time/space.

This is, in spirit, an analogue of the Church–Turing Thesis, transplanted into the realm of geometric computability. Can any non-VN-like abstract machine compute something that can be computed by a VN-like abstract machine and vice versa? As with the Church–Turing Thesis, it is best regarded as a conjecture—a foundational principle rather than a theorem. Its true justification comes from the robustness of computation models, not from formal proof.

Conjecture 5.2 (Geometric Curry–Howard–Lambek Correspondence — Symbolic Layer)**.** *For every Turing machine TM there exists a symbolic GL and a symmetric monoidal functor:*

$$E : \; \mathsf{Conf}(TM) \;\longrightarrow\; \mathsf{Path}(G \ltimes M)$$

from the free category of TM configurations and one–step rewrites to the path category of the GL action groupoid, such that:

(a) *E is full and faithful on traces (strong bisimulation up to stuttering);*

(b) *for any symmetric monoidal semantics $Z : \mathsf{Bord}_n^S \to \mathcal{CTO}$ extending GL, there is a natural isomorphism*

$$Z \circ E \;\cong\; [\![-]\!]_{TM}$$

in $\mathcal{CTO}$ (same denotation of computations).

This is a more genuine correspondence of the Curry-Howard Type. The study of homotopy computation will one day help illuminate this issue, and again we should remember that all topological spaces have these homotopies via the fundamental groupoid, but most seem to lack the correct geometric symmetry for actual computation, compared to transcendental curves. Algebraically, however, we are in a territory that is only beginning to be explored.

5.1 Implications for Reversible Computing

GCT opens an avenue for *reversible computing*[45]. In reversible computing, computation is analogous to the laws of physics, running both backwards and forwards without loss of information, a desirable property. Data is not overwritten like the tape on a Turing Machine. More formally, a reversible computation is a functor

$$\mathcal{F} : \mathcal{C} \to \mathcal{C} \; \exists x \mid \mathcal{F}^{-1} : \mathcal{C} \to \mathcal{C}$$

with

$$\mathcal{F} \circ \mathcal{F}^{-1} = \mathrm{id}\mathcal{C}$$

In the case of group-based computing, the symbolic-only mode, this means every computation is a bijection on the state space, that is, an automorphism. In group-theoretic terms:

$$\forall g \in G, \exists g^{-1} \in G \text{ such that } gg^{-1} = e.$$

Thus, computation becomes a path through a groupoid where partially recursive functions are enabled, supporting the claim that information is never fundamentally lost, only transformed. One can imagine a flow of fluid going in one direction, but suddenly reversing in another back to its original state. For example, one can picture a fluid flow reversing direction and restoring its initial configuration,

or a THM in which unit helices exchange twist factors and then "rewind" their evolution, returning the system to its starting state via composed reversible operations.

Let $\mathcal{M}$ be a GCM. Let:

$$\phi_t : M \to M$$

be the flow map from a vector field or group action, where $t \in \mathbb{R}$ is time. If ϕ_t is a diffeomorphism for each t, and

$$\phi_{-t} = \phi_t^{-1}$$

then the evolution law is classified as reversible.

5.2 Comparison to the Blum-Shub-Smale Model

A foundational precedent for my approach can be found in the Blum–Shub–Smale (BSS) model [46] of computation, which extends classical computability theory from the discrete domain of $\mathbb{N}$ into the continuous field $\mathbb{R}$. In the BSS model, the computational machine ("BSS machine") operates over the real numbers, with registers that hold real values, and instructions that perform real arithmetic or conditional branching over those values. It defines recursive functions over $\mathbb{R}$, enabling a theory of complexity and decision problems in $\mathbb{R}$-space.

However, the BSS machine is fundamentally algebraic and analytic in its structure. It is built over field operations and assumes the availability of real-number arithmetic at unit cost. This decision leads to a rich mathematical structure, but one that remains semantically symbolic, not physical. In contrast, GCT builds a theory of computation grounded in topological and geometric structure, not just symbolic algebra. While a BSS machine generalizes the register machine, a GCM generalizes the state space itself. That is, the BSS machine retains a fixed architecture and extends the data, while a GCM lattice changes the nature of computation itself by encoding operations directly into the geometry and deformation of manifolds. Computation arises not from the manipulation of symbols or numbers, but from the controlled evolution of geometric state.

Definition 5.1 (Geometric E_∞-algebra and Non-Von Neumann Hypercomputation)**.** Let $\mathcal{C}_\infty$ denote the symmetric monoidal ∞-category of smooth (derived) manifolds and smooth maps, equipped with the Cartesian product $\times$ as its tensor product. A GCM lattice is specified by:

$$X \in \mathcal{C}_\infty, \qquad \left\{\mu_n : X^{\times n} \longrightarrow X\right\}_{n \geq 0},$$

where the operations μ_n encode parallel composition (disjoint gluing) of n unit helix components. The family (μ_n) is required to satisfy the coherence relations of the E_∞-operad in $\mathcal{C}_\infty$, making $(X, \mu_\bullet)$ a commutative E_∞-algebra object is as follows:

(a) The unit map $\mu_0 : * \to X$ corresponds to the empty (null) computation.

(b) The binary product $\mu_2 : X \times X \to X$ realizes the geometric join of two independent lattice components.

(c) Higher μ_n encode n-way concurrent composition, commutativity holds up to coherent homotopy, giving the required flexibility for fluidic deformations.

In this language, a classical BSS machine is recovered by restricting to the full sub-∞-category of finite–dimensional real vector spaces with polynomial maps, whose objects form discrete E_∞-algebras (strict commutative monoids). The GCM lattice generalizes this: it is still an E_∞-monoid, but now geometry (i.e., smooth structure, group actions, bordisms) replaces uncountable real registers as the primitive data store.

$$\text{BSS (discrete commutative monoid)} \quad \longrightarrow \quad \text{GL (geometric } E_\infty\text{-algebra).}$$

See Lurie [47] for the general theory of E_∞-algebras in symmetric monoidal ∞-categories. This synthesis of higher category theory and geometry moves us closer to a physical theory of computation, where space, time, and shape are no longer external to computation, but computation itself.

5.3 Relationship to Topological Kleene Field Theories and Bordism Framings

Miranda, Peralta, and Gonzales [48] have recently introduced Topological Kleene Field Theory (TKFT), which aims to advance understanding of fluidic computation beyond classical Turing completeness.

At the heart of TKFT is the notion of a *smooth dynamical bordism*—a manifold connecting multiple disk boundaries, each embedded with a Cantor set of natural numbers, within which computation unfolds as a flow governed by partial recursive (reaching) functions. In this framework, computation is realized as trajectories in a Hilbert space, and the "pair of pants" topology elegantly encodes program branching, granting TKFT both expressive power and a high degree of abstraction.

However, the GCMs central to our theory do not naturally inhabit n-dimensional Hilbert spaces. This raises a conceptual question (echoing the cobordism hypothesis): how should we interpret "bordism" when the manifold itself is the computational substrate, rather than a medium mediating between external boundaries?

Definition 5.2 (Computational Geometric Bordism)**.** Let M_0, M_1 be (oriented) n-dimensional GCMs. A *computational geometric bordism* from M_0 to M_1 is a compact smooth $(n+1)$-manifold W with boundary equipped with a decomposition $\partial W \cong \overline{M_0} \sqcup M_1$ (and choice of collars), together with any specified flow/group/action data used for computation. The bordism W is the geometric realization of the computational transformation from M_0 to M_1.

Example 5.1 (The Helix as a Geometric Bordism). Recall that a bordism between two closed n-manifolds M_0, M_1 is an $(n+1)$-manifold W with boundary: $\partial W = M_0 \sqcup M_1$. If we embed the unit circle S^1 into $\mathbb{R}^2$ by the usual trigonometric map:

$$(\theta, t) \longmapsto \big(\cos(\theta + \lambda t),\ \sin(\theta + \lambda t),\ t\big), \qquad (\theta, t) \in S^1 \times I.$$

and then thicken this embedding by the interval $I = [0, 1]$, we obtain

$$W = S^1 \times I \subset \mathbb{R}^3, \qquad (\theta, t) \longmapsto \big(\cos\theta,\ \sin\theta,\ t\big).$$

The boundary components $S^1 \times \{0\}$ and $S^1 \times \{1\}$ are two disjoint copies of the circle, and the cylinder W is a smooth $(1+1)$-dimensional bordism connecting them.

A helix is a cylinder twisted by a chirality map: $(\theta, t) \mapsto (\cos\theta,\ \sin\theta,\ t + \lambda\theta)$, so its core curve winds through space while still furnishing an $(n+1)$-manifold whose boundary is the same pair of n-manifolds. Thus, a helix (or any smooth chiral flow) is a geometric computational bordism: the shape itself is the intermediary manifold realizing the cobordism relation.

In GCT, a more intrinsic and lightweight perspective is natural. Here, bordism is *internal* to the GCM: computation is realized as deformation or flow between different boundary states within the manifold itself. The "input" and "output" boundaries correspond to loci where topological flows implement morphisms. As currently formulated, TKFT operates at a computational power comparable to our FHCM—the quasi-fluidic, maximally abelian regime of geometric computation. TKFT is constructed over countable topological fields ($\mathbb{N}$) and exploits the full power of recursive and automata-theoretic computation, and can recursively evolve its computational history, formulated over countable topological fields. But it does not natively capture continuous phase/flow decorations of our fully-powered analog $SO(2)$ regime.[25]

If TKFT aspires to capture the laws of fluidic computation in terms of a general-purpose "CPU" architecture—capable of universal computation via smooth dynamical flows—then GCT can be viewed as a more discrete, helical "GPU": a specialized, fuzzy, and highly efficient processor for geometric tasks. So we do not wish for persons to construe GCT as not a competitor to TKFT, but a complementary framework, with distinctive strengths and tradeoffs. In principle, both approaches can be used in concert to address fundamental problems in topological physics from different vantage points.

It will be fascinating to observe how these approaches rooted in TQFT, TKFT, and now GCT, develop in parallel, potentially converging to a deeper, unified mathematics of fluidic computation.

[25] It's best for readers to understand the BSS machine as a kind of "transition point" between Turing Machines and Pour-El-and-Richards-style geometric chaos.

5.4 Towards the Computational Cobordism Hypothesis

The algebraic enrichment required for this framework demands categories capable of encoding both geometric and computational structure. To this end, we introduce the category $\mathcal{ESM}^+$ ("Embedded Smooth Manifolds with Enrichments") as the natural habitat for all GCMs.

Definition 5.3 (Embedded Smooth Manifolds with Enrichments)**.** Let **Man** denote the category of smooth manifolds and smooth maps. The *master category* $\mathcal{ESM}^+$ is a symmetric monoidal category whose:

(a) **Objects** — are pairs (M, str) where M is a smooth (possibly stratified) manifold and str is a finite tuple of auxiliary geometric/computational structures over M (e.g. an orientation double cover, specified principal bundles and connections, distinguished vector fields/flows, discrete labels, etc.)

(b) **Morphisms** — $(M, \mathrm{str}) \to (N, \mathrm{str}')$ are smooth maps $f : M \to N$ that preserve the declared structure (pull back covers/bundles, intertwine flows, respect discrete labels.)

(c) **Monoidal product** — is disjoint union:

$$(M, \mathrm{str}) \otimes (N, \mathrm{str}') := (M \sqcup N,\ \mathrm{str} \sqcup \mathrm{str}'), \qquad 1 := (\varnothing, \varnothing).$$

There is a faithful forgetful functor $U : \mathcal{ESM}^+ \to \mathbf{Man}$. We suppress set-theoretic size issues (all categories are essentially small).

Remark 5.1. The choice $\otimes = \sqcup$ models parallel composition of independent systems; other monoidal products (e.g. fibered sums under boundary matching) can be introduced when needed, but are not required for the present development.

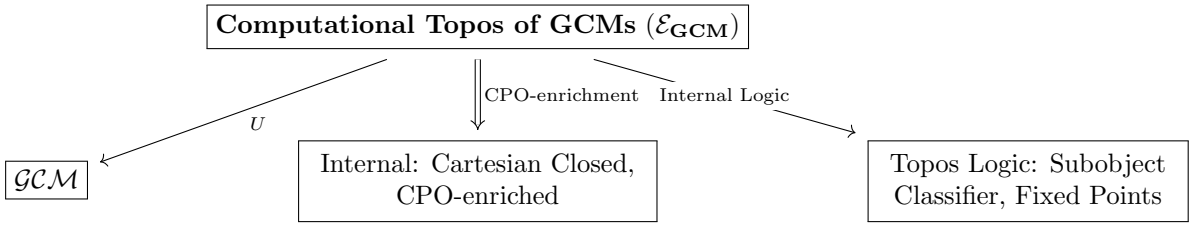
We separate the geometric substrate from its semantic evaluation target.

Definition 5.4 (Computational Topos of GCMs)**.** A *computational topos* $\mathcal{CTO}$ is an elementary topos equipped with a symmetric monoidal structure (taken to be coproduct $\sqcup$ unless stated otherwise), together with a faithful interpretation functor:

$$\mathsf{Eval} : \ \mathcal{ESM}^+ \longrightarrow \mathcal{CTO}$$

that preserves finite limits and the declared monoidal structure on objects and morphisms. Being a topos, $\mathcal{CTO}$ is cartesian closed and admits a subobject classifier; thus, it carries an internal intuitionistic higher-order logic in which we reason about programs and states encoded by decorated groupoids and bordisms.

To ensure the existence of computational fixed points within the categorical framework, we require the category to be both cartesian closed and enriched with a domain structure. Specifically, complete partial orders (CPOs) on its subobjects. This combination supports fixed-point theorems (Kleene), essential for modeling recursion and partial computations.



Remark 5.2 (CPO Enrichment for Recursion). To model partiality and fixed points (Kleene/Scott), one may assume a reflective subcategory $\mathcal{CTO}_{\mathrm{cpo}} \subseteq \mathcal{CTO}$ whose objects carry complete partial orders and whose morphisms are Scott-continuous. Then standard fixed-point theorems apply to endomaps in $\mathcal{CTO}_{\mathrm{cpo}}$. The enrichment details are orthogonal to the geometric layer and can be developed independently.

We now pass from the state/transition layer (decorated groupoids) to a control–flow layer where computations are realized as decorated bordisms.

Definition 5.5 (Decorated Bordism Category, Once-Extended)**.** Fix a dimension $n \geq 0$, a decoration monoid/group S, and a class of $(n{+}1)$–dimensional compact smooth manifolds with boundary and S–decoration data compatible with composition (gluing). The *decorated n–bordism category* Bord_n^S has:

(a) **Objects** — are closed $(n{-}1)$–manifolds with chosen S–decorations on their state/transition groupoids (e.g. $d : \Pi_1^*(\,\cdot\,) \to \mathbf{B}S$).

(b) **Morphisms** — are compact n–manifolds W with boundary $\partial W \cong \overline{M_0} \sqcup M_1$ carrying compatible S–decorations, composed by gluing along collars.

(c) **Symmetric monoidal products** — are given by disjoint union.

When a geometric substrate $(M, \mathrm{str}) \in \mathcal{ESM}^+$ is fixed, we write $\mathsf{Bord}_n^S(M, \mathrm{str})$ for the full subcategory of bordisms that embed in $M \times I$ and whose decorations agree with str on the boundary.

Conjecture 5.3 (The Computational Cobordism Hypothesis (CCH), Informal Version)**.** *Let $\mathcal{ESM}^+$ be the ambient category, and let $\mathcal{CTO}$ be a computational topos. There is a classification principle such that symmetric monoidal functors*

$$Z \;:\; \mathsf{Bord}_n^S(M, \mathrm{str}) \longrightarrow \mathcal{CTO}$$

*(*geometric field theories*) are in natural bijection with choices of* fully dualizable *decorated objects associated to (M, str) in a suitable symmetric monoidal $(\infty, 1)$–completion of $\mathcal{ESM}^+$. Moreover, when S is trivial and decorations factor through Π_1, this reduces to the corresponding case of the (once-extended) Cobordism Hypothesis.*

Three–line operational sketch.

(a) **States/transitions** — would encode local updates by an S–decorated (groupoid or action groupoid) $(\mathcal{G}, d)$ over boundary cuts.

(b) **Control–flow** — models a computation as a decorated bordism $W \in \mathsf{Bord}_n^S(M, \mathrm{str})$; composition = concatenation of phases.

(c) **Evaluation** — is a symmetric monoidal functor Z sends boundary data to objects of $\mathcal{CTO}$ and sends W to a morphism in $\mathcal{CTO}$, yielding denotational semantics for the computation.

Remark 5.3 (Reduction to Classical Cases)*.* If the decoration d factors through Π_1 and S is discrete/finite, then Z specializes to a finite–group TQFT. If S carries continuous abelian data (e.g. $S^1 \times C_3$), Z gives an abelian "gauge–like" semantics; non-abelian S interpolates to gauge–type models.

We now pass from the state/transition layer (decorated groupoids) to a control–flow layer where computations are realized as decorated bordisms.

Example 5.2 (Helical Bordism Computation)*.* Begin by taking a $U(1)$-valued 1-cocycle on $S^1 \times D^2$.

Let $M \cong S^1_\theta \times D^2_{(r,\varphi)}$ with chirality $\chi \in C_3$ and an orientation phase $\theta \in S^1$. Take:

$$G \;=\; C_3 \times SO(2) \times \mathbb{R}, \qquad S \;=\; S^1 \times C_3, \qquad \alpha(\varepsilon, R_\alpha, t) \;=\; \big(e^{i\alpha},\, \varepsilon\big).$$

Then G acts on the core state manifold by: $(\varepsilon, R_\alpha, t) \cdot (\chi, \theta, \phi, \dot\phi) = (\varepsilon\chi, \theta + \alpha, \Phi_t(\phi, \dot\phi))$, and the action groupoid $G \ltimes M$ admits the decoration d_α. Consider the trivial bordism (a cylinder) $W = S^1 \times I \subset M \times I$ with the helical embedding:

$$(\theta, t) \longmapsto \big(\cos(\theta + \lambda t),\, \sin(\theta + \lambda t),\, t\big), \quad (\theta, t) \in S^1 \times I,$$

whose boundary circles are $S^1 \times \{0\}$ and $S^1 \times \{1\}$. Along a loop at fixed radius r, define a local angular rate $\Omega(r, \theta)$ and the one–turn phase gate:

$$U(r) \;=\; \exp\Big(i \int_0^{2\pi} \Omega(r\theta)\, d\theta\Big) \in S^1.$$

The decorated groupoid assigns $d([\gamma_r]) = \big(U(r), \chi\big)$ to the loop; two loops with the same homotopy class but different radii $r_1 \neq r_2$ may yield $U(r_1) \neq U(r_2)$. Thus, the computational outcome depends on geometric control (rather than on the bare π_1–class), yet composition in S remains abelian and compatible with the groupoid structure.

A *geometric field theory* $Z : \mathsf{Bord}_1^S(M, \mathrm{str}) \to \mathcal{CTO}$ sends the boundary circle to an object $Z(S^1) \in \mathcal{CTO}$ and the decorated cylinder W to a morphism $Z(W)$ which, in this example, acts by multiplication by $U(r)$ on the appropriate component (and flips chirality if $\varepsilon = -1$). When Ω is constant and χ is trivial, d factors through Π_1 and Z reduces to a discrete finite–group semantics.

Lest the scope of this paper appear overwhelming, we should all be well aware that the exploration of the Computational Cobordism Hypothesis will be a difficult, decades-long project, but one that will hopefully put to rest some of the deepest mysteries in our universe once and for all.

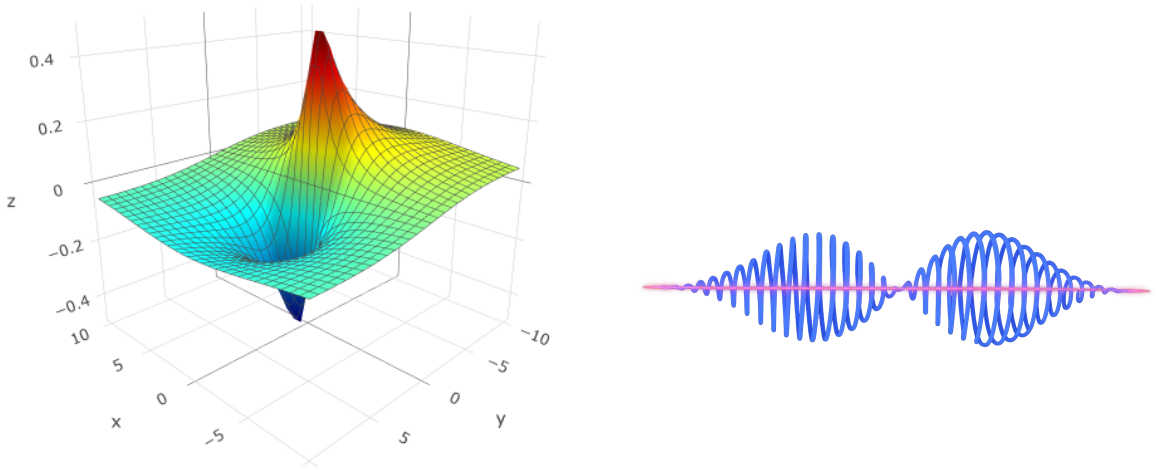
5.5 Computing the Uncomputable

As a short aside, it is natural to ask whether quasi-fluidic computational models might transcend the limits of classical computation, such as those demarcated by the Busy Beaver function [49]. However, as long as the system remains deterministic and locally finite, the Busy Beaver remains unassailable. Its uncomputability is a structural, not architectural, phenomenon. True transcendence of computability, if it even exists, lies only in the fluidic regime, which leaves the discrete computational universe altogether and eventually transcends into the realm of outright gauge symmetries. So while the Busy Beaver function represents the ultimate barrier for computable functions in classical, discrete models, both a BSS machine equipped with a sufficiently powerful oracle and a GCM operating in the fully fluidic regime can, in principle, transcend this barrier. See Aaronson [50] for an updated perspective on uncomputability.

Hence, to surpass Turing completeness is not merely to extend symbolic computation, but to explicitly address problems known from Pour-El and Richards's phenomenon: physically relevant PDEs whose solutions, though perfectly deterministic and mathematically consistent, are nonetheless beyond classical Turing-computable methods. Geometric computation, by embedding computations within continuous geometric structures, provides a deterministic and mathematically rigorous framework capable of directly representing and solving exactly these non-computable functions.

In this sense, Geometric Computability Theory may be viewed as a kind of long-lost cousin to Geometric Complexity Theory, but one that focuses instead on physics rather than algorithmic complexity. Thus, the apparent boundary of classical computation becomes a new frontier, opening up the rigorous computational exploration of quantum systems, field theories, and deep mathematical problems like the Yang–Mills mass gap.

5.6 Constructive Computational Gauge Theory



(a) BPST instanton: a finite-action, localized topological solution to the Yang–Mills equations, exhibiting geometric twist and winding.

(b) Complex quantum wave: a superposition of oscillations, illustrating the intertwined, helical modes of computation in the geometric framework.

Figure 6: Some Interesting Examples of Geometric Configurations Corresponding to Type 3 and Type 1 THM.

The area between mathematics and physics has been a great source for fertile cross-pollination of ideas with work from mathematicians like Kontsevich [51], Connes [52], and physicists like Witten [53] tackling questions related to quantization, fundamental force symmetries, and spin, to name but a few.

Helices are plentiful in the field of modern physics. As an example, for any solution to the free-particle Schrödinger equation, THM Type 1 can be interpreted as the real and imaginary components, intertwined, with the joint at points of equal phase:

$$\Psi(x,t) = Ae^{i(kx-\omega t)}$$

$$\Re[\Psi(x,t)] = A\cos(kx-\omega t)$$

$$\Im[\Psi(x,t)] = A\sin(kx-\omega t)$$

Plotting against the x-axis, we get a helical system spiraling through space. The two components are phase-shifted by $\pi/2$, so one spirals left, the other right, relative to the phase progression forming the "helical orb" in 6 resembling the Type 1 THM. Helices appear in a number of other areas. To name but a few: photon polarization is a classic example, Dirac and Weyl spinors have a geometric representation as helices on the Bloch sphere, and more recently, skyrmions have been studied as manifesting as topologically protected twisted and helical states [54]. These are not just simple coincidences, but instead speak to the deep geometrical roots of quantum phenomena.

Recent and highly invaluable work by Bliokh et al. [55] has demonstrated that the surface Maxwell modes at media interfaces are governed by a topological invariant valued in $\mathbb{Z}_4$. Their approach, rooted in phase winding, represents a major advance in our understanding of topological invariants in photonic systems.

However, in my view, this classification does not fully account for the independent binary symmetries present in the physical configuration—specifically, the left/right and up/down combinations encoded in a THM_4 construction. While the choice of $\mathbb{Z}_2 \times \mathbb{Z}_2$ as an invariant is logical within their framework, a reformulation in terms of the Klein four-group V_4 more accurately reflects the underlying symmetry structure. This viewpoint directly parallels the base 2-geobit state space, providing both a more granular symmetry classification and a richer foundation for both physical insight and computational applications.

Remark 5.4 (Dual Helicity Join Structures). It should be noted that there exists a fundamental duality between helices joined at their bases and those joined at their apexes. In the context of geometric computation, we define THM manifolds as those helices that join smoothly at their bases, forming a continuous manifold with no topological singularity. These are designed for deterministic logic flow.

By contrast, helices joined at an apex are familiar from physical models of spin, Dirac operators, and helicity transport. Such structures are inherently singular at the join point and correspond to sources, sinks, or field singularities—examples include optical vortices, Dirac cones, or other topological defects.

This duality is summarized below:

Feature	**Twin Helical Manifolds**	**The Apex Twin Helix**
Join point	Smooth base join	Singular apex join
Topology	Globally smooth manifold	Manifold with point singularity
Flow	Bidirectional	Radial (source/sink)
Purpose	Logical/computational flow	Field excitation, spin transport
Chirality	Controlled by local geometry	Emerges from singularity
Singularities	Avoided (non-singular)	Essential (Dirac monopoles)
Field role	Symbolic/geometric logic	Physical spinor dynamics
Interpretation	Topological logic circuits	Helicity-driven field structures

This duality suggests that helicity can play two conceptual roles: in geometric computation, smoothness enables information flow, but in topological physics, singularity encodes conserved charges.

As established throughout this work, helices exhibit exceptionally rich geometric behavior. The pre-quantum computational geometry of twin helices, characterized by the expressive potential of geometric flows and deformations, may thus provide a substrate from which quantum phenomena—such as the discrete state spaces of spin foam models—naturally emerge. Rather than postulating quantum discreteness as a primitive, it is reasonable to view quantum structure as arising as an invariant or stable phase of the underlying computational dynamics, much as solitons and topological defects emerge from continuous systems. So under geometric computation, quantum spin networks

arise via coarse-graining of the underlying micro-geometry of helices. The helix's intrinsically limited symmetry enables highly efficient storage and propagation of data, but as a side effect of its wild "macro-expressive" geometry, it eventually leads to quantum behavior. Hence, it requires the all-too-familiar n-dimensional Hilbert space to capture this "computational phase space" of possibilities.

Remark 5.5. Although this paper did not originally aim to engage in ongoing debates over the unification of physics, it is worth noting that the geometric computation model developed here vindicates the combinatorial and categorical perspectives advocated by proponents of Loop Quantum Gravity. In addition, it closely aligns with the non-Markovian view of quantum mechanics espoused by physics philosopher Jacob Barandes [56].[26] In this paper's somewhat heterodox view [27], discrete quantum geometry emerges as a natural phase transition within a more general, continuous geometric logic—without the need to discard the underlying manifold structure.

Remark 5.6. While the constructions in this paper have been of the more categorical variety, it would be amiss not to note that twistor theory, pioneered by Penrose and Rindler [57], achieves a stunning algebraic encoding of geometric and conformal symmetries in four dimensions, providing powerful tools for the study of massless field equations and scattering amplitudes. GCT encompasses twistors as its fundamental limit case: they appear precisely where the state space of computation exhibits continuous ω-logic, and the algebraic closure group is maximally abelian and fully continuous (e.g., $SO(2)$). It should be clarified that real-life twistors do not engage in actual real-valued infinite computation, as this would violate Bekenstein's bound [58]. Hence, it's more of an infinitesimal computational evolution, a kind of "synthetic hypercomputation" using smooth nilpotent infinitesimals. Thus, twistors, the generators of spacetime, become a distinguished phase within a broad universe of transcendental, computational geometric structures—a special case of the general principle that geometry and computation are deeply intertwined.

Furthermore, the computational framework described here strengthens the connection between the Geometric Langlands program and twistor theory, originally suggested by Scholze and discussed by Woit [59]. In particular, the use of deterministic geometric computations over real-valued continuous structures enables the explicit realization of twistor correspondences and geometric Langlands dualities as computational phenomena. Thus, the relationship between number-theoretic structures and geometric/analytic objects can, and certainly will, be directly validated and explored computationally within this unified real-valued framework.[28]

Remark 5.7 (Geometric Computation as a Language for QG). We should frame quantum gravity, whether loop or string-based—and more generally gauge theory—as a problem of expressiveness versus verifiability. If we allow "all histories" (arbitrary geometries, topology change, gauge redundancy, unbounded recursion in the construction of spacetimes), amplitudes become ill-defined and intractable. If we clamp down too hard, we lose physically relevant states and dynamics. **Functional programming offers a blueprint for balancing these extremes**. My proposal is a constructive computational gauge theory that strikes a principled middle ground: a typed, linear, total, effect-controlled calculus of geometries. Concretely, boundary data (3-geometries with gauge labels) are the types; spacetime regions (4-dimensional histories/cobordisms) are the terms; and gluing is composition. This gives a compositional semantics familiar from TQFTs but designed to scale beyond the purely topological setting (i.e., Chern-Simons).

Readers are asked to consider the correspondences in Table 11. Three design choices enforce computability and physics: linearity, totality, and effects. Linearity tracks boundary degrees of freedom as conserved resources (no cloning/erasure), so unitarity and charge conservation are built into the typing discipline rather than imposed post hoc. Totality means the "geometry evaluator" (our state-sum/variational executor) always normalizes: amplitudes exist and are finite in the core fragment. The phenomena that usually force uncontrolled manipulations such as coarse-graining, stochastic mixing,

[26] While certain cellular automata interpretations of quantum mechanics lean towards superdeterminism, GCT rejects this; measurement settings are treated as exogenous boundary data; at the ontic level this yields measurement dependence via a global, time-symmetric constraint, while operational free choice and no-signaling are preserved by the linear/gauge symmetries.

[27] If people are to insist on a separate term for this thesis I would suggest "Geometric LQG." That said, Sir Penrose has spoken recently with Mr. Jaimungal about the need to "gravitize" the Standard Model rather than vice versa, so how "heterodox" GLQG is remains open for debate.

[28] I must confess this is (yet again) ironic: *Complexity and Real Computation* opens by referencing Penrose—not on physics, but for fractals. Yet both twistors and the BSS model are, in reality, the prime examples of continuous algebraic computational flow!

Table 11: Programming Concepts and Quantum Gravity Analogues

Programming Concept	**Quantum Gravity Analogue**
Types	Boundary states (3-geometries with gauge data)
Terms / Programs	4-geometries (cobordisms, histories)
Composition	Gluing of spacetime regions
Linear types	Conservation laws, unitarity (no-cloning of boundary data)
Totality	Termination of the "geometry evaluator" (finite amplitudes)
Effects & handlers	Coarse-graining and renormalization
Dependent types	Gauge and diffeomorphism constraints

and renormalization are modeled explicitly as algebraic effects with handlers. In this way, renormalization becomes a controlled transformation of programs, not an ad hoc subtraction. Dependent types encode gauge and diffeomorphism constraints at the level of well-typedness, so invariances propagate mechanically through compositions.

Within this calculus, amplitudes are evaluations, symmetries live in the types, and RG/coarse-graining are effect handlers. The proposed helical primitives provide the concrete generators of histories: smooth, orientable flows that carry discrete topological labels (orientation/chirality) alongside continuous geometry. This marries the "continuous versus discrete" tension: spectra and curvature are continuous objects; quantum numbers arise as stable, counted winding data. Practically, the workflow is: specify typed boundary data; assemble regions from helical primitives; compose; evaluate; and, where needed, apply effect handlers that implement scale changes with proofs of soundness.[29]

The payoff is a language that is *expressive* enough to describe nontrivial gauge dynamics and background independence, yet *restricted* enough to prove normalization, locality/compositionality, and anomaly-freeness in the core. Extensions (matter content, topology change, nonperturbative sectors) are added modularly as new effects or controlled type extensions, preserving verification theorems as we widen scope. In short: constructive computational gauge theory provides a semantics where we can calculate, compose, and certify. This shifts the idea of well-behaved QFT/QG from "internet math folklore" to a "usable, checkable substrate." For the foundational work on constructive quantum field theory, see Baez, Segal, and Zhou [11]. Our approach here is in this spirit, but computational.

Remark 5.8 (On Calabi and Geometric Flows). This paper is admittedly not done in the intellectual paradigm of modern string theory, but we note that Eugenio Calabi's vision of canonical geometric structures obtained via analytic deformation and geometric flow laid the groundwork for much of modern geometry and geometric analysis. While the famous Calabi conjecture was originally doubted even by S.-T. Yau (who went on to prove it), its legacy endures: the study of geometric flows, moduli spaces, and canonical metrics remains at the heart of mathematical physics. The present framework *tries* to fulfill a vision in line with Calabi's geometric philosophy. In this sense, a "geometry as computation" paradigm may be viewed as a natural generalization of Calabi's original vision.

Calabi-Yau manifolds would arise in GCT as distinguished points or subspaces within the broader moduli space of possible structures. They correspond to exceptionally symmetric, stationary, or integrable solutions—the "harmonic gates" of the configuration space. While essential for certain regimes (notably, the lower abelian or quasi-fluidic phases), they represent rare, fine-tuned configurations rather than the generic case. In this sense, Calabi-Yau manifolds function as "gatekeepers." They maintain the special coherency and integrability in the computational landscape, keeping the threat of non-abelianism at bay.

Remark 5.9 (Disclaimer Regarding the Relationship to Entropic Gravity and the Lorentzian). Lest the framework proposed be misinterpreted, it should be specified that geometric computation works at the pre-metric substrate and derives causal/optical structure via an effective constitutive tensor χ. In near-equilibrium regimes χ becomes metric-induced, recovering Lorentzian geometry and Prof. Schuller's pre-metric electromagnetism as an effective semantics. On larger scales, the thermodynamic equation of state reproduces Prof. Jacobson's derivation of Einstein's equations. So Schuller's and

[29] Einstein was *right* to distrust non-functorial quantization and primitive probabilities, and the helix shows us *why*. In our model, the beable is the geometric flow, outcomes live in a symbolic sheaf, quantization is a structured (local, gluable) functor, and probabilities are pushforwards of invariant measures—not axioms.

Jacobson's perspectives appear as complementary limits of a single constructive framework: EM fixes the light cone via χ; coarse-graining of the underlying computation promotes χ to $\chi[g]$ and yields GR. No contradiction is implied—only a layering of descriptions. Explicit validation of the relationship between entropic gravity and the underlying chaotic computational dynamics implied by Jacobson's work [60] will be a natural consequence of the advancement of this framework over time.

Likewise, we do not quantize gravity in the traditional sense. We do not treat the metric as another Yang–Mills field and force a continuum path integral over geometries. Quantum features (probabilities, interference) arise from structured coarse-graining of deterministic geometric programs, while GR appears as the large-scale limit of sound gluing rules. Here, gravity is not "quantized" as a field; it is the constraint architecture of the computation, with quantum behavior emerging from how we compress and observe it.

As time is constrained, the example of a potential application of GCT we would now like to highlight would be that of the infamous Yang-Mills mass gap and existence. The examples listed below illustrate how certain nontrivial solutions to the Yang–Mills equations and related field theories are, in fact, naturally helical:

Calorons: Periodic instantons in gauge theory [61].
Sphalerons: Unstable saddle points in electroweak theory [62].
Twisted solutions: Solutions over compactified or cylindrical spacetimes [63].
Witten's helical solutions: Helical structures in certain lower-dimensional reductions [64].
Non-abelian plane wave solutions: Chiral, twist-like structures in non-abelian gauge theory [65].

While this discussion does not claim to exhaust the intricate connections between geometric computation and modern field theory, as illustrated in Figure 6a, one can see that the gluing of two helical manifolds at a common boundary can form a topological structure directly analogous to the BPST instanton of Yang–Mills theory. It is here that we now enter into the final of the four computational regimes, *emergent gauge*.

Each unit helix manifold H_A, H_B can be modeled as open sets $U_A, U_B \subset M$, each equipped with a local trivialization of the associated bundle—that is, local coordinate patches where the fiber structure is simple. The gluing of two helices at their base corresponds to identifying the bundles over the intersection $U_A \cap U_B$ via a transition function $g_{AB} : U_A \cap U_B \to G$. In gauge theory, such a function is interpreted as a gauge transformation; geometrically, it is a diffeomorphism on the base and a G-action on the fibers. Deformations and transitions between these glued manifolds encode discrete topological information, serving as geometric analogues of bit-flip operations. Thus, each unit helix in a GCM can be interpreted as a topological soliton, and computation in this framework is deeply connected to the instanton sector of gauge field theory. This positions geometric computability as the natural foundation for a "computational Yang–Mills" framework—one that treats a GCM as a fiber bundle in low-dimensional topology, enabling the construction of a mathematical theory of *computational gauge fields*.

$$\begin{array}{ccc} \text{Local Helix Chart } (U_A) & \xrightarrow[\text{Transition Function}]{g_{AB}} & \text{Local Helix Chart } (U_B) \\ \downarrow & & \downarrow \\ \text{Bundle Trivialization } P|_{U_A} & & \text{Bundle Trivialization } P|_{U_B} \end{array}$$

The diagram above encodes the configuration space of these gluings: the resulting moduli space is precisely the structure used in Yang–Mills theory to describe instantons. Thus, geometric computation in this regime is naturally understood as computation within the moduli space of topological solitons. For the foundational theory on the geometry of Yang-Mills fields, see Atiyah [66]; for a modern perspective, see Donaldson [67].

Example 5.3 (Minimal Bundle–Connection Data). Let $P \to M$ be a principal G–bundle with local trivializations over $U_A, U_B \subset M$ and a transition map $g_{AB} : U_A \cap U_B \to G$. A connection is given by local 1–forms $A_A \in \Omega^1(U_A, \mathfrak{g})$ and $A_B \in \Omega^1(U_B, \mathfrak{g})$ satisfying the standard patching on overlaps:

$$A_B \;=\; g_{AB}^{-1} A_A \, g_{AB} \;+\; g_{AB}^{-1} dg_{AB}, \qquad F_B \;=\; g_{AB}^{-1} F_A \, g_{AB},$$

with curvature $F_A = dA_A + A_A \wedge A_A$. When the computation is represented as a bordism in time, we work on $M \times I$ with temporal gauge; the BPST sector corresponds to $G = \mathrm{SU}(2)$ and ASD curvature on $M \times I$:

$$F \ + \ *F \ = \ 0.$$

The topological charge (instanton number) on a compact 4–manifold $\widehat{M}$ (compactification of $M \times I$) is:

$$k \ = \ \frac{1}{8\pi^2} \int_{\widehat{M}} \mathrm{tr}\,(F \wedge F) \in \mathbb{Z}.$$

In the abelian/quasi–fluidic limit $G = U(1)$, a loop $\gamma \subset M$ carries the computational phase:

$$d_\tau([\gamma]) \ = \ \mathrm{Hol}_\gamma(A) \ = \ \exp\Big(i \int_\gamma A\Big), \qquad d_\chi \in C_2 \text{ for chirality,}$$

and these labels transform covariantly under a gauge change g_{AB}. Gluing two helices along $U_A \cap U_B$ modifies the transition function $g_{AB} \mapsto h\, g_{AB}$ and hence the induced holonomies on linking loops (that is, a bit–flip by transition). In the SU(2) sector, a computational step can be modeled as an ASD connection on $M \times I$; the boundary Chern–Simons functional changes by an amount governed by k, providing a topological "gate" in the gauge regime.

Hence, in the geometric computation framework, abelian solutions correspond to the stable storage and propagation of data—periodic, commutative, easily reversible. Non-abelian solutions correspond to the transformation of data: their flow actively encodes logic gates, information processing, and state transitions that cannot be reduced to simple, commutative propagation. The highest regime—emergent gauge—allows for history-dependent, topological computation, where even the process of transitioning between states (as in sphalerons—see Manton [68]) becomes part of the logic. This paper's inevitable conclusion that the Standard Model is, in all likelihood, a buggy computer made of helices is outlined in 12.[30]

Table 12: Geometric Computation Regimes and The Equations of Motion

Abelian Regimes		**Non-Abelian Regimes**	
Symbolic	**Quasi-Fluidic**	**Fluidic**	**Gauge**
Discrete logic states, stable storage	Periodic or cyclic state transitions, memory elements	Data dynamically transformed by non-commutative flows	Topological transitions, history-dependent computation
Classical EM fields, digital bits	Calorons, periodic instantons, "clock" solutions	Non-abelian plane waves (Coleman), helical/twisted solutions (Witten)	Sphalerons, instantons, tunneling between vacua

Example 5.4 (The Computational Consequences of Non-Abelian Plane Waves). As a motivating example of one solution cited above, consider the appearance of non-abelian plane waves as described by legendary Harvard physicist Sidney Coleman:

$$L = L_{\text{extr.}} = -\frac{1}{4}\mathrm{tr}(F_{\mu\nu}F^{\mu\nu}) = \frac{1}{2}\sum_{a=1}^{n}\left[(E^a)^2 - (B^a)^2\right] =: \frac{1}{2}(E^2 - B^2).$$

This is discussed by Kastrup in [69] where he notes (with some exasperation) that when electric and magnetic energy densities are equal, $E^2 = B^2$. This signals more than a mere mathematical curiosity. In the context of geometric computation, these solutions have profound consequences. Unlike their abelian counterparts, non-abelian plane waves possess internal "twist" and holonomy: as the field propagates, its configuration traverses nontrivial paths in the gauge group manifold. This means the solution itself acts as a dynamic, information-carrying process capable of encoding group-theoretic state transitions as it evolves. Kastrup's observation that $L = 0$ defines a bifurcation hypersurface in field space directly parallels the notion of a logic gate or switching surface in computational theory.

[30] As to why the Universe chooses to *express itself* this way? The author kindly asks that questions be directed elsewhere. The answer is probably in one of Sir Penrose's books I haven't read yet, but, unlike him, I am not yet qualified to opine.

In this geometric framework, the set of such plane wave solutions forms a universal "signal alphabet," providing the basic computational flows (or gates) within the field theory. Hence, the Yang–Mills equations do not merely describe static states, but embody a rich logic of state transitions—realized physically as flows through field configuration space. This connection justifies treating these classical solutions as concrete "geometric logic gates": they demonstrate that the fabric of non-abelian gauge theory naturally supports computational processes, with the group structure supplying the logic and the geometry supplying the flow. Nature is *expressive*, she just likes shortcuts.

Earlier in this paper, an executive decision was made to work within $\mathbb{R}^4$, rather than remaining in $\mathbb{R}^3$, thereby abandoning Type 2 THM and focusing on the subgroups corresponding to Type 1, 3, and 4 THM.[31] Recall that the official statement is:

> *Prove that for any compact simple gauge group G, a non-trivial quantum Yang–Mills theory exists on $\mathbb{R}^4$ and has a mass gap $\delta > 0$.*

$\mathbb{R}^4$ is equipped with its standard Riemannian (Euclidean) structure, as required by the context of the mass gap problem. GCMs, being smooth manifolds with boundary, naturally inherit a Riemannian structure. While discrete gauge theories have been explored for decades, as Jaffe and Witten note, they remain insufficient for a full resolution of the Yang-Mills problem [70].

Remark 5.10. While the physical Yang–Mills mass gap is posed over Minkowski spacetime $\mathbb{R}^{1,3}$, my model operates in a purely spatial 4-manifold, using the fourth spatial dimension to encode extended chirality-resolving structure (e.g., the Type 2 Twin Helix). Time evolution in our system is replaced by global pulses or flow propagation, yielding an internal clocking structure without invoking temporal metrics.

Remark 5.11 (Comparing Lattice Gauge Theory and GCM Lattices). Lattice gauge theory provides a discretized model of gauge field dynamics, using an external lattice as a computational scaffold. While it preserves local symmetries and allows for numerical simulation, the continuum limit remains problematic, and computability is not intrinsic to the mathematical structure. By contrast, a GCM lattice encodes its state space, symmetry, and allowed transitions intrinsically via computability principles.

The "lattice" arises from the manifold's own group-theoretic and topological constraints, ensuring that spectral gaps and forbidden transitions are a direct consequence of the system's geometry, not an artifact of discretization. Thus, a GL could serve as a fundamentally computable model of gauge field dynamics, where geobits play the role of discrete, intrinsic gauge degrees of freedom. This means the quantum (or information-theoretic) behavior of geometric computation is encoded not by a measure on configurations, but by the structure of the sheafified moduli space. In this setting, path integration is replaced by the act of sheafification within a Grothendieck topos.[32]

Remark 5.12 (Issues with Hyper-Turing Complete Computation Over $\mathbb{R}$). It is at this point, however, a problem we have so far put aside rears its head—modeling the computation. Recall table 10. As one approaches the full-bore fluidic computation needed to model the Yang-Mills equations in a simple Lie group, typical notions of computability (Kleene) effectively break down. In this limit, one must appeal to topological, cohomological, or quantum invariants to encode information, and the notion of computation becomes vastly more abstract.

In principle, one might attempt to implement the gauge regime using a highly expressive fluidic GCM defined over $\mathbb{R}$. However, as the BSS model illustrates, computation over the reals is notoriously difficult to control. For this reason, I've emphasized synthetic rather than classical differential geometry as the appropriate framework in the two uppermost regimes.

Although the halting problem can, in some sense, be decided in unrestricted real computation models, this requires introducing non-constructive operations—such as branch functions that implicitly encode oracle calls. Even then, computations that are trivial over $\mathbb{N}$ (e.g., equality testing) can become undecidable over $\mathbb{R}$. Moreover, certain gaps or spectra may only be accessible via limiting processes or transcendental solutions that cannot be finitely encoded. If the only way to compute or even define the mass gap involves non-recursive reals, then the problem is, in effect, nonconstructive and incomputable. The BSS framework shows that real computation without restrictions yields a model of hypercomputation that is vastly more powerful, but also pathological from the standpoint of classical recursion theory. This leads to a sad possibility:

[31] The lesson here is if something only works in 4D, that's generally a good sign, not a bad one.

[32] "It's nice to have a framework where you can only have one Lagrangian, but it would be even nicer to have some reason that this particular algebra would inevitably appear as an answer to some question." -Edward Witten [71]

Conjecture 5.4 (A Lost Melody for the Yang–Mills Mass Gap)**.** *Let $\mathcal{S}$ be the solution space of (quantum) Yang–Mills fields over $\mathbb{R}^4$, as formulated in the Clay Millennium Problem. Suppose the mass gap $\Delta > 0$ exists: that is, there exists an energy difference between the vacuum and the first excited state.*

Then, in the setting where $\mathcal{S}$ is modeled as a space of real-valued (Schwartz, Sobolev, or distributional) functions:

(a) *There exists a real number r_Δ encoding the value of the mass gap such that r_Δ is* ***recognizable*** *(in the sense of infinite time or real computation: BSS/ITTM machines), but not* ***computable*** *(i.e., there is no finite algorithm—classical or transfinite—that outputs r_Δ).*

(b) *The existence of such a mass gap is, up to isomorphism, a "lost melody" real: there is no effective procedure to construct or compute the gap, but its value can be "verified" if presented as input.*

(c) *The recognizability–noncomputability boundary in the Yang–Mills mass gap is* ***isomorphic****, in the sense of descriptive set theory and infinite computation, to the phenomenon described by the Lost Melody Theorem for BSS/ITTM machines.*

(d) *In particular, if the Yang–Mills mass gap is posed over the full continuum, then the problem is undecidable in the constructive sense. Any constructive proof must restrict the solution space to computable, algebraic, or otherwise "tame" settings.*

I conjecture that the Yang–Mills mass gap problem, as posed over the full continuum of real-valued field configurations, is isomorphic to the Lost Melody Theorem in infinite computation: its solution may be recognizable but not computable. That is, the mass gap is a "lost melody" real—provable to exist in principle, but forever unconstructible by algorithmic means. This suggests that the problem is undecidable in $\mathbb{R}$, and that a constructive solution is only possible in a suitably tamed mathematical universe.

Originally, the author hesitated to include this conjecture, but given these limitations, we must fully accept, and it would be unethical for me not to disclose, that even if the entire mathematical apparatus were developed to attempt such a calculation, it may not be possible to model mass gaps in $\mathbb{R}$. The limitations discovered by BSS, and before that, Turing and Gödel, still haunt the boundaries of the mathematical universe. Our new tools bring us closer, but cannot guarantee victory in all cases. It may be a song that can be heard, but never fully sung.

Remark 5.13. Full treatment of the Lost Melody Theorem is beyond scope here; see Carl [72] for a recent proof on the ITBSS model and Hamkins and Lewis [73] for the original background. The theorem is completely unknown outside the study of specialized computational logic, but a result verifying this conjecture would not be without precedent—see Cubitt et al. [74] for a quantum treatment on the undecidability of spectral gaps using lattice Hamiltonians demonstrating that certain physical properties in quantum many-body systems are algorithmically inaccessible. Similar "lost melody" effects could be intrinsic to classical geometric and dynamical systems. This extends the reach of computability barriers beyond quantum models, suggesting that the undecidability pervades both quantum and classical physics, rooted in the deep structure of field and manifold theory itself.

Still, more optimistically, in Yang–Mills we speak of four "constants" (the three gauge couplings and θ_{QCD}). Operationally, they are measured to finite precision. For a computable formulation, it therefore suffices to treat parameters and intermediate quantities as effectively representable numbers (e.g., rationals, algebraic numbers, or computable reals), rather than invoking the full uncountable continuum $\mathbb{R}$. This is not a novelty—see, e.g., Szudzik's observation that physical models do not require arbitrary real numbers to be expressed or simulated [75] (in addition, his entire thesis is, of course, highly relevant to us). The stance is pragmatic: the continuum remains a powerful idealization, but the algorithms by which we build and evaluate amplitudes only need number systems that admit effective approximation and decision procedures.

The point is not that "$\mathbb{R}$ is wrong," but that computability depends on the base field. Choosing a constructive base lets us keep the familiar analysis when needed while ensuring every step of the formalism is executable. In particular, nothing in the approach depends on sub-Planckian claims; we simply avoid making uncountable structure a primitive of the dynamics. A high-level road map toward the Yang-Mills Mass Gap is outlined below:

(a) **Define Computational Semantics for Gauge Theory** — Define a typed, linear, total calculus whose types encode boundary data (3-geometries with gauge labels) and whose terms encode histories (4-dimensional "programs" composed by gluing).

(b) **Build Computable Geometric Bundles** — Give the calculus a denotational semantics as a computable fiber-bundle (connection, parallel transport) formulated in a constructive/synthetic differential setting over a computable base field. This keeps differential structure while ensuring all primitives are effectively evaluated.

(c) **Assess the Scaling Limit to Continuum Yang-Mills** — Prove that families of programs approximate continuum Yang–Mills in a controlled scaling limit. Concretely: a "differential computational jet calculus" provides error bounds showing convergence of correlation functions/Wilson loops to those of continuum Yang-Mills as the program lattice is refined.

(d) **Prove the Gap and its Stability** — Establish a spectral gap (uniform lower bound on excitation energies) for the constructive model using the calculus' linear/compactness properties; then show this lower bound survives the scaling limit, yielding a continuum mass gap. Here, the constructive setting replaces global non-compact searches with local-to-global, type-theoretic invariants.

All physically relevant predictions are recoverable with parameters and observables taken from an effective number system (e.g., computable reals). This keeps the continuum as a limit, not as a prerequisite, and lets "proof by construction" coexist with the analytic tools physicists use.

This approach reframes the Yang–Mills mass gap problem as a question of information-theoretic structure. Can the dynamics of a gauge field theory, defined over the continuum and acted on by a compact simple Lie group, enforce a nontrivial discrete gap in its computational state space? This translation from an energy gap to an information gap offers a new mental rotation for understanding both the foundations of gauge theory and the limits of computability in physical systems. **In essence, to attempt to solve the mass gap, we cannot just try to compute the solutions *to* the Yang Mills equations, but rather we may have to compute *with* the solutions to the equations themselves.**

To my knowledge and research, this perspective is absent from the current literature, and its development may offer novel tools for bridging mathematics, physics, and information theory. Needless to say, the final step of the outline would involve an excruciatingly complex derivation process. The actual group needed to "power" such a computational process would likely be none other than $\mathbf{Diff}(\mathbf{M})$ itself—the group of all diffeomorphisms on smooth manifolds. This group is, in a sense, an "algebra that contains all geometry." Recall the chart of n-logics from 7. This is the final entry at the top. Its geometric expressiveness—"Extreme," its algebraic closure—"Maximal." To compute with it is to model the entire universe of possible smooth invertible maps in every dimension. To program with $\mathbf{Diff}(\mathbf{M})$ is to wield an omniscient power over the very fabric and shape of space itself. By executing the bargain with Atiyah's devil in reverse, you gain near limitless geometric power, but at a terrible algebraic cost.

Yet, all things considered, and given how little progress has been made thus far, this approach may still be worth pursuing, despite the peril.

Remark 5.14 (Disclaimers Regarding the Nature of THM). While speculative connections between THM_4's structure and fundamental particle physics or gauge interactions may seem tempting, we do not believe this to be the case. The most adventurous but conservative approach we should take towards physical interpretation currently is through discrete gauge-like symmetries. The V_4 group does show up several times in the context of Yang-Mills. Specifically, the four fundamental geometric twin helices invariants may naturally encode discrete internal symmetries analogous to parity (P) and time-reversal (T) transformations. In particular, it is *possible* that T is connected to helix orientation and P is connected to helix chirality.

Indeed, there is something suggestive about the Klein four-group; it is the minimal abelian structure where every non-identity is its own inverse. Historically, EPR highlighted a real incompleteness under locality and separability; Bell showed that package is incompatible with quantum correlations. A deterministic, globally constrained model relaxes measurement independence while preserving no-signalling, so Einstein, Podolsky, and Rosen's [76] impulse toward a deeper description is retained, but the specific locality/separability question would receive a negative answer.

Our universe may be built from the smallest of groups, not just the richest ones.

Disclaimers

Regarding Fundamental Assumptions

The results presented herein **should under ABSOLUTELY NO CIRCUMSTANCES be interpreted as a challenge** to experimentally validated quantum field theories (such as Quantum Chromodynamics, Yang–Mills theory, or the Standard Model). Rather, they provide a rigorous geometric and deterministic framework offering new conceptual insights into symmetry, gauge structure, and hidden-variable formulations. This work aims to *enrich* our understanding of existing physics, not supplant or refute it.

Author's Note on Naming Conventions

Given that THM may turn out to be a fundamental building block of our universe/gauge theory, **I strongly feel that it should not bear anyone's name**, if this is the case. I understand why the naming of manifolds was done with Calabi-Yau, but in this case, my gut says doing the same is *probably* a very bad idea.

I would, however, not mind if the actual computational lattice structure (Kaminsky Lattice) were named after me. I get that my wishes are unconventional, but this is more so what I *don't* want named after me, rather than what I do, and so I simply ask that it be respected, given the unusual circumstances.

That said, the focus of this paper is not personal legacy, but the advancement of geometric computability and its physical and pedagogical applications.

Funding Sources

None. If you wish to support this work, my Ko-Fi is: ko-fi.com/benjaminsky

Conflicts of Interest

None.

AI Contributions and Acknowledgments

Google Gemini and OpenAI ChatGPT were used to assist in generating GAP code and Sage code for 3D THM models and geobit space states, as well as to automate the more labor-intensive aspects of LaTeX typesetting, in lieu of graduate student support. I claim full responsibility for all ideas, results, and ramifications presented herein. Many productive conversations were had with AI regarding the more technical aspects of Yang–Mills theory, CPT symmetry, and fluidic type theory, which informed my writing and perspective.

Due to the excruciating technical difficulty of Babai's groundbreaking quasi-polynomial Graph Isomorphism algorithm, multiple LLMs were used to validate my algorithmic approach (loosely based on Babai's method) in concert with each other.

AI is a powerful tool, but as Sir Penrose notes, only if one can ask the right questions—which, in the end, is what mathematics is all about.

References

[1] Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, New York, NY, 1 edition, 1978.

[2] Hassler Whitney. The self-intersections of a smooth n-manifold in $2n$-space. *Annals of Mathematics*, 45(2):220–243, 1944.

[3] Matthew Cook. Universality in elementary cellular automata. In Andrew Northern and Paola Carrasco, editors, *Proceedings of the First Annual Wolfram Science Conference 2003*, Champaign, IL, 2004. Wolfram Media.

[4] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[5] M. Aschbacher. On the maximal subgroups of the finite classical groups. *Inventiones Mathematicae*, 76:469–514, 1984.

[6] Richard Phillips Feynman. *Feynman Lectures on Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, United States, 1998.

[7] John C. Baez and John W. Barrett. The quantum tetrahedron in 3 and 4 dimensions, 1999.

[8] John C. Baez. Getting to the bottom of noether's theorem, 2022.

[9] Matthias Felleisen. On the expressive power of programming languages. *Science of Computer Programming*, 17(1-3):35–75, 1991.

[10] Tai-Danae Bradley. At the interface of algebra and statistics, 2020.

[11] John C. Baez and James Dolan. Higher-dimensional algebra and topological quantum field theory. *Journal of Mathematical Physics*, 36(11):6073–6105, November 1995.

[12] F. William Lawvere. Diagonal arguments and cartesian closed categories. *Lecture Notes in Mathematics*, 92:134–145, 1969.

[13] M. Atiyah. Mathematics in the 20th century. *Bulletin of the London Mathematical Society*, 34:1–15, 2002.

[14] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[15] Robert Cardona, Daniel Peralta-Salas, and Albert Smith. Constructing turing complete euler flows in dimension 3. *Proceedings of the National Academy of Sciences*, 118(19):e2026818118, 2021.

[16] Terence Tao. On the universality of the incompressible Euler equation on compact manifolds. *Discrete and Continuous Dynamical Systems*, 38(3):1553–1565, 2018.

[17] brusspup. Amazing water and sound experiment #2. YouTube, 2013. Accessed: 2025-07-01.

[18] TB Thompson, P Chaggar, E Kuhl, A Goriely, and Alzheimer's Disease Neuroimaging Initiative. Protein-protein interactions in neurodegenerative diseases: A conspiracy theory. *PLoS Computational Biology*, 16(10):e1008267, Oct 2020.

[19] A. Jadbabaie, N. Motee, and M. Barahona. On the stability of the kuramoto model of coupled nonlinear oscillators. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 4296–4301 vol.5, 2004.

[20] Marian Boykan Pour-El and Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90, 1979.

[21] Peter M. Kogge. *The Architecture of Symbolic Computers*. McGraw-Hill, New York, 1991.

[22] G.J. Sussman, J. Holloway, G.L. Steel, and A. Bell. Scheme-79 — lisp on a chip. *Computer*, 14(7):10–21, 1981.

[23] R. C. Penner. The decorated Teichmüller space of punctured surfaces. *Communications in Mathematical Physics*, 113:299–339, June 1987.

[24] Brendan Fong. Decorated cospans, 2015.

[25] John C. Baez, Kenny Courser, and Christina Vasilakopoulou. Structured versus decorated cospans. *Compositionality*, 4:3, September 2022.

[26] Andreas Floer. Morse theory for lagrangian intersections. *Journal of Differential Geometry*, 28(3):513–547, 1988.

[27] Ralph L. Cohen. Floer homotopy theory, revisited, 2019.

[28] Daniel Müllner. Orientation reversal of manifolds. *Algebraic and Geometric Topology*, 9(4):2361–2390, November 2009.

[29] Stephen Smale. On the structure of manifolds. *American Journal of Mathematics*, 84(3):387–399, 1962.

[30] Nadia Chouaieb, Alain Goriely, and John H. Maddocks. Helices. *Proceedings of the National Academy of Sciences*, 103(25):9398–9403, 2006.

[31] Toshiaki Adachi. Foliation on the moduli space of helices on a real space form. *International Mathematical Forum*, 4(35):1699–1707, 2009.

[32] Gino Loria. *Spezielle algebraische und transscendente ebene Kurven. Theorie und Geschichte.* B. G. Teubner, Leipzig, 1902.

[33] László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.

[34] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *CoRR*, abs/1812.01537, 2018.

[35] Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Computer Science: Faculty Publications, Smith College*, 2005.

[36] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 171–183, New York, NY, USA, 1983. Association for Computing Machinery.

[37] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC '19, page 217–228. ACM, June 2019.

[38] Arvind and David E. Culler. Tagged Token Dataflow Architecture. Memo 229, Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group, jul 1983. Also presented at EASCON '83, Washington D.C., September 19, 1983.

[39] Seth A. Steinberg, Don Allen, Laura Bagnall, and Curtis Scott. The butterfly™ lisp system. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI '86, Philadelphia, PA, 1986. AAAI Press.

[40] P. J. Landin. The next 700 programming languages. *Commun. ACM*, 9(3):157–166, March 1966.

[41] Robert S. Smith, Eric C. Peterson, Mark G. Skilbeck, and Erik J. Davis. An Open-Source, Industrial-Strength Optimizing Compiler for Quantum Programs. *arXiv preprint arXiv:1910.09033*, 2019. Rigetti Computing technical report, last updated April 1, 2020.

[42] Jacob Lurie. On the classification of topological field theories. *Current Developments in Mathematics*, 2008:129–280, 2009.

[43] Ulrik Buchholtz and Johannes Schipp von Branitz. Primitive recursive dependent type theory, 2024.

[44] S. K. Donaldson. An application of gauge theory to the topology of 4-manifolds. *Journal of Differential Geometry*, 18:279–315, 1983.

[45] Kenichi Morita. *Theory of reversible computing*. Springer, 2017.

[46] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-COMPLETENESS, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1), jul 1989.

[47] Jacob Lurie. *Higher Algebra*. 2017. Available at https://www.math.ias.edu/~lurie/papers/HA.pdf.

[48] Ángel González-Prieto, Eva Miranda, and Daniel Peralta-Salas. Topological Kleene Field Theories: A new model of computation. *arXiv preprint arXiv:2503.16100*, 2025.

[49] Tibor Radó. On non-computable functions. *The Bell System Technical Journal*, 41(3):877–884, May 1962.

[50] Scott Aaronson. The busy beaver frontier. *The American Mathematical Monthly*, 127(9):790–798, 2020.

[51] Maxim Kontsevich. Deformation quantization of poisson manifolds, I. *Letters in Mathematical Physics*, 66(3):157–216, 2003.

[52] Alain Connes. Non-commutative geometry and the Standard Model of elementary particle physics. *Journal of High Energy Physics*, 2000.

[53] Edward Witten. Quantum field theory and the Jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, 1989.

[54] Naoto Nagaosa and Yoshinori Tokura. Topological properties and dynamics of magnetic skyrmions. *Nature Nanotechnology*, 8(12):899–911, 2013.

[55] Konstantin Y. Bliokh, Daniel Leykam, Max Lein, and Franco Nori. Topological non-Hermitian origin of surface Maxwell waves. *Nature Communications*, 10, 2019.

[56] Jacob A. Barandes. The stochastic-quantum correspondence, 2023.

[57] R. Penrose and M. A. H. MacCallum. Twistor theory: An approach to the quantisation of fields and space-time. *Physics Reports*, 6(4):241–315, 1973.

[58] Raphael Bousso. The holographic principle. *Rev. Mod. Phys.*, 74:825–874, Aug 2002.

[59] Peter Woit. Notes on the twistor $\mathbf{p}^1$, 2022.

[60] Ted Jacobson. Thermodynamics of spacetime: The einstein equation of state. *Physical Review Letters*, 75(7):1260–1263, August 1995.

[61] T. C. Kraan and P. van Baal. Periodic instantons with non-trivial Holonomy. *Nuclear Physics B*, 533(1-3):627–659, 1998.

[62] F. R. Klinkhamer and N. S. Manton. A saddle-point solution in the Weinberg-Salam theory. *Physical Review D*, 30(10):2212–2220, 1984.

[63] G. 't Hooft. A property of electric and magnetic flux in nonabelian gauge theories. *Nuclear Physics B*, 153:141–160, 1979.

[64] E. Witten. Topological Quantum Field Theory. *Communications in Mathematical Physics*, 117(3):353–386, 1988.

[65] S. Coleman. Nonabelian plane waves. *Physics Letters B*, 70(1):59–60, 1977.

[66] M. Atiyah. Geometry of Yang-Mills fields. In G. Dell'Antonio, S. Doplicher, and G. Jona-Lasinio, editors, *Mathematical Problems in Theoretical Physics*, volume 80 of *Lecture Notes in Physics*. Springer, Berlin, Heidelberg, 1978.

[67] S. K. Donaldson. Yang-mills theory and geometry. Unpublished article, Imperial College, London, January 2005.

[68] N. S. Manton. The inevitability of sphalerons in field theory. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 377(2161):20180327, November 2019.

[69] H. A. Kastrup. Canonical theories of lagrangian dynamical systems in physics. *Physics Reports*, 101(1-2):151, 1983.

[70] Arthur Jaffe and Edward Witten. Quantum Yang–Mills Theory. In James Carlson, Arthur Jaffe, and Andrew Wiles, editors, *The Millennium Prize Problems*, pages 129–152. Clay Mathematics Institute, 2000.

[71] Int'l Centre for Theoretical Physics. Conversation: Salam, sciama, witten and budinich, Oct 2014. Accessed: 2025-06-18.

[72] Merlin Carl. The lost melody theorem for infinite time blum-shub-smale machines, 2020.

[73] Joel David Hamkins and Andy Lewis. Infinite time turing machines, 1998.

[74] Toby Cubitt, David Perez-Garcia, and Michael M. Wolf. Undecidability of the spectral gap. *Forum of Mathematics, Pi*, 10, 2022.

[75] Matthew P. Szudzik. *The Computable Universe Hypothesis*, page 479–523. WORLD SCIENTIFIC, October 2012.

[76] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47(10):777–780, 1935.