Geometric Computability: Twin Helices and the Algebra of Flow

Benjamin Kaminsky

July 9, 2025

Abstract

In this work, I construct a manifold comprising of twin helices that explicitly validates the existence of computational geometric bordisms, as anticipated in the broader program of the cobordism hypothesis. I investigate its geometric, topological, algebraic and computational properties. This work suggests the emergence of a new research direction that I call "Geometric Computability Theory", situated at the crossroads of geometry, topology, and computability theory. I prove that encoding of arbitrary size finite state machines is possible using these "Geometrically Computable Manifolds" leading to the emergence of Turing Complete phenomena. Further constructions are made on the nature of quantum-like and n-ary/fuzzy computing behavior leading to the formation of a general metatheorem for the behavior of computational logic to explain the origin of Rice's Theorem. I describe their behavior in terms of Kleene's theory of partial recursive functions. Following that I use ordinary differential equations to model discrete fluidic computation. This perspective opens numerous avenues for future investigation, potentially reshaping our understanding of computation in continuous spaces including applications to the infamous Yang-Mills existence and mass gap. As of writing, algebraic systems to formalize the behavior these geometrically computable manifolds appears to be absent. Future work will involve the development of formal algebraic frameworks, rigorous computational models, and experimental simulation of system behaviors.

Contents

1 Introduction

The following paper is admittedly difficult to write an introduction for, try as I might.

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.

The model emerges from an 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 encodings with differential-geometric ones, specifically, moduli spaces of helices and their categorical behavior under symbolic flow.

From this construction arises a new class of computational objects: deterministic symbolic flows that admit non-invertible, non-cancellative monoidal structure due to chirality constraints, and whose torsion data encode the computational equivalent of randomness within otherwise reversible systems. The resulting enriched category naturally supports symbolic computation with bounded state and fixed-point behavior, and exhibits expressive capacity that surpasses classical VM-like models (e.g., Turing machines and register machines) while retaining verifiability at the topological level.

Along the way, the paper draws from and in some cases unifies results from differential geometry, Lie groupoids, and topological quantum field theory. In particular, the construction directly validates predictions made in the work of Baez (notably on the cobordism hypothesis and quantum tetrahedra) and synthesizes ideas from Miranda's work on topological semantics and Felleisen's work on language expressiveness. The outcome is a model that sits at the intersection of computability, topology, and symbolic representation and in doing so, subtly suggests the inadequacy of current homotopy-theoretic assumptions for describing such enriched systems.

Notably, while the construction is purely deterministic, it reveals 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. The paper concludes 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.

There is little else I can say at this point, except if you are to read it, do so with care, and a few times over—if need be. The extreme interdisciplinary approach is by necessity. Part of me wishes it was not so for the sake of accessibility, but changing any element would diminish the impact of what I wish to communicate about this model. Due to the the nature of the subject, I've purposely chosen explanation with heavy reliance on analogization maps, charts, and worked examples. This is a feature—not a bug. Papers addressing foundational computational models are rare, ranging from the technically austere (Turing, Shannon, Cook) to the more recent and abstract (Blum-Shub-Smale, Miranda-Peraltes-Cardonez).

There is exactly one Lagrangian in this paper. The rest is computation. There is no standard blueprint for writing a paper that defines an entire computational model from scratch. The number of people in history who have defined such computational models is a vanishingly small one. I now find myself in the rather strange position of being among them.

I think the nature of mathematical discovery is all to often shrouded, much like the subject of my study. So, as I pull back the curtain, so too can you join me.

Thank you.

2 FVISA Solids and the Properties of Twin Helical Manifolds

2.1 Replacing the Horn

Let us begin by supposing that an evil topology demon from a higher-dimensional plane has invaded our mortal $\mathbb{R}^3$ world.

Being a naturally mischievous creature, it has taken away our ability to rotate asymptotic $\mathbb{R}^2$ curves into $\mathbb{R}^3$, in an attempt to silence the archangel Gabriel by erasing his horn. Students in calculus classes across the world can no longer learn about the canonical paradoxical example of solids of finite volume and infinite surface area ("FVISA") using surface integrals.

Can we still generate a solid with the FVISA property to teach them? Thankfully, the answer is yes. By using helical, rather than rotational, symmetry, such an object is within constructive reach. In fact, not only is it attainable, but its inherent algebraic and computational richness will become apparent as we proceed.

We construct a helical solid with FVISA properties 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. The cross-sectional radius is given by a parametric function $\rho(\theta)$, where

$$\rho(\theta) = \frac{k}{\theta}$$

and a and k are positive constants $k > 0$. We consider the domain $\theta \geq \theta_{\min}$ for some $\theta_{\min} > 0$ to avoid the singularity at $\theta = 0$. This function inflates the spine of our helix, assigning it volume over all points in its domain. As the radius decays, $\rho(\theta)$ normally produces a helical solid of finite surface area, but in our case we adjust it to achieve the FVISA condition.

Definition 2.1. Let $\rho(\theta) = \frac{k}{\theta}$ define the cross-sectional radius of the helix for θ in some domain $[\theta_{\min}, \theta_{\max}]$, with $k > 0$.

Base: The base of the helix is defined as the point where $\rho(\theta)$ attains its maximal value, i.e., at $\theta = \theta_{\min}$:

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

Equivalently, given a desired base radius $\rho_{\text{base}} > a$, the corresponding value of $\theta_{\min}$ is

$$\theta_{\min} = \frac{k}{\rho_{\text{base}}}.$$

Apex: The apex is defined as the point where $\rho(\theta)$ attains its minimal value, i.e., at $\theta = \theta_{\max}$:

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

2.2 Volume and Surface Area Integrals on Helical Solids

For the solid formed by extending a tubular neighborhood around the helical spine, the volume V and surface area SA are given by the following integral formulas. Consider a tube that extends infinitely along the spine for $\theta \in [\theta_{\min}, \infty)$. Then:

$$V = \int_{\theta_{\min}}^{\infty} \pi[\rho(\theta)]^2 \, |\gamma'(\theta)| \, d\theta$$

$$SA = \int_{\theta_{\min}}^{\infty} 2\pi\rho(\theta)\,|\gamma'(\theta)|\,d\theta$$

Here, $|\gamma'(\theta)|$ is the magnitude of the tangent vector to the curve $\gamma(\theta)$, which corresponds to the differential arc length element along the central spine.

For our standard helix

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

the derivative is

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

whose magnitude is constant:

$$\begin{aligned} |\gamma'(\theta)| &= \sqrt{(-R\sin\theta)^2 + (R\cos\theta)^2 + h^2} \\ &= \sqrt{R^2 + h^2}. \end{aligned}$$

We denote this constant arc length element by $L_\gamma = \sqrt{R^2 + h^2}$.

Substituting $|\gamma'(\theta)| = L_\gamma$ and $\rho(\theta)$ into the formulas above yields:

$$V = \pi L_\gamma \int_{\theta_{\min}}^{\infty} \left(\frac{k}{\theta}\right)^2 d\theta \quad \text{and} \quad SA = 2\pi L_\gamma \int_{\theta_{\min}}^{\infty} \left(\frac{k}{\theta}\right) d\theta.$$

To ensure the FVISA property, we examine the asymptotic behavior of the cross-sectional radius function $\rho(\theta)$. In general, $\rho(\theta)$ must decay toward zero as $\theta \to \infty$ in a manner resembling a power law. For instance, let:

$$\rho(\theta) = \frac{C}{\theta^q}, \quad \text{with } C > 0,\ q > 0$$

Then the corresponding helical solid exhibits FVISA behavior *if and only if* the exponent q lies in the range

$$\frac{1}{2} < q \leq 1.$$

This ensures that the total volume converges while the surface area diverges due to the slow decay of $\rho(\theta)$.

For symbolic integration, let us take the specific case $\rho(\theta) = \frac{k}{\theta}$. Then the integrals simplify to:

$$\begin{aligned} V &= \int_0^{\infty} \frac{\pi k^2 \sqrt{R^2 + h^2}}{\theta^2}\, d\theta = \pi k^2 \sqrt{R^2 + h^2} \\ SA &= \int_0^{\infty} \frac{2\pi k \sqrt{R^2 + h^2}}{\theta}\, d\theta \to \infty. \end{aligned}$$

Thus, we have constructed a geometric object with the FVISA property, achieved not by rotating 2D curves, but through helical extension and decay.

Remark 2.1. For our purposes, we define a valid output from $\rho(\theta)$ for a non-infinite helix as satisfying two requirements:

1. It must describe a closed curve forming a smooth 2-manifold.
2. The value at $\rho(\theta_{\max})$ must not produce a point singularity at the apex of the helical solid.

Regarding the second requirement, although it is not immediately relevant, every finite helix constructed in this paper will remain topologically smooth at both endpoints. This is accomplished by using a non-linear function for $\rho(\theta)$, such as a squared sine profile:

$$\rho(\theta) = \rho_{\max} \sin^2\left(\pi \frac{\theta - \theta_{\text{start}}}{\theta_{\text{end}} - \theta_{\text{start}}}\right)$$

for $\theta \in [\theta_{\text{start}}, \theta_{\text{end}}]$, and $\rho(\theta) = 0$ otherwise. This produces a tube that smoothly tapers to zero radius at both ends, with vanishing derivative at the endpoints and non-singular tips.

For parametric simplicity, we select the circle as the volumetric cross-section for all computations in this section. However, the requirement to avoid singularities at the helix apex can be satisfied in many ways. Topologically, it suffices to ensure smoothness via any appropriate diffeomorphism near the endpoints, even if the cross-sectional radius remains constant elsewhere.

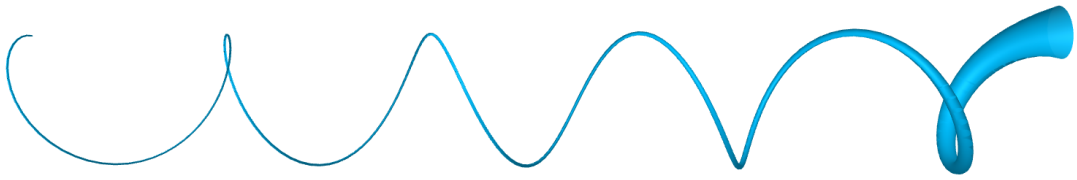


Figure 1: A New FVISA Solid. Its Sound Profile Differs From That of Gabriel's Horn.

2.3 Chirality and Alignment

Definition 2.2. 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 g such that $g(O) = r(O)$.

Definition 2.3. 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 $h > 0$.

Lemma 2.1 (Spiral Matching Lemma)**.** *Let S_1 and S_2 be Archimedean spirals. A C^2-smooth curve connecting S_1 to S_2 at a common point P exists if and only if the position, tangent direction, and curvature of S_1 and S_2 are all equal at P.*

Proof. The conditions stated in the lemma—matching position, tangent, and curvature—are the standard requirements for ensuring C^2 continuity between any two curves at a junction point. The "if" part of the statement is true by definition. We focus on proving the "only if" part, which demonstrates that these conditions must hold and leads to a constraint on the spirals' chirality.

Let the Archimedean spirals S_1 and S_2 be described in polar coordinates by the equations $r = a_1\theta$ and $r = a_2\phi$, respectively. The constant a determines both the spread and the *chirality* (or handedness) of the spiral. By convention, if $a > 0$, the spiral is counter-clockwise (left-handed), and if $a < 0$, it is clockwise (right-handed).

For a C^2-smooth transition, the signed curvature κ_s of both spirals must be equal at the point of connection, P. The signed curvature of an Archimedean spiral $r = a\theta$ is given by:

$$\kappa_s(\theta) = \frac{\theta^2 + 2}{a(\theta^2 + 1)^{3/2}}$$

Crucially, the sign of the curvature is determined by the sign of the parameter a. A left-handed spiral ($a > 0$) has a strictly positive signed curvature, while a right-handed spiral ($a < 0$) has a strictly negative signed curvature.

Assume, for the sake of contradiction, that S_1 and S_2 have opposite chiralities but can be joined at a point P by a C^2-smooth curve. If the chiralities are opposite, then their defining parameters a_1 and a_2 must have opposite signs. This implies that their signed curvatures, $\kappa_{s,1}(P)$ and $\kappa_{s,2}(P)$, must also have opposite signs.

The condition for a C^2 join is $\kappa_{s,1}(P) = \kappa_{s,2}(P)$. However, if two non-zero numbers have opposite signs, they cannot be equal. The only way for this equality to hold would be if both curvatures were zero.

$$\kappa_{s,1}(P) = \kappa_{s,2}(P) = 0$$

Observing the curvature formula, the numerator $\theta^2 + 2$ is never zero for any $\theta \in \mathbb{R}$. Thus, the curvature of an Archimedean spiral is never zero. This is a contradiction.

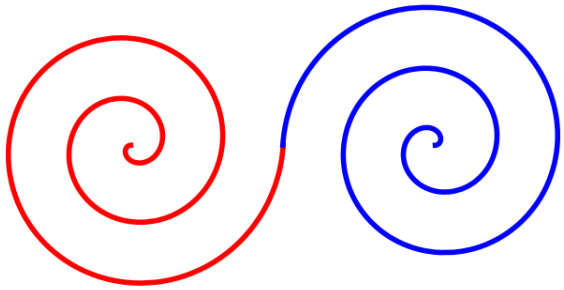


Figure 2: Same chirality alignment among spirals, a "top down" view for our helices

Therefore, a C^2-smooth connection cannot be formed between two Archimedean spirals of opposite chirality. For the curvatures to be equal at P, the spirals must have the same chirality (i.e., $\text{sgn}(a_1) = \text{sgn}(a_2)$). □

Corollary 2.2 (Spiral Chirality Constraint)**.** *Let S_1 and S_2 be Archimedean spirals. A C^2-smooth path connecting them at a common point can exist only if their chiralities are the same.*

Proof. This is a direct consequence of the Spiral Matching Lemma. As proven above, the requirement for matching curvatures at the connection point, $\kappa_{s,1}(P) = \kappa_{s,2}(P)$, can only be satisfied if the signs of the curvatures are the same, which in turn requires the spirals to have the same chirality. □

Remark 2.2. This lemma illustrates a fundamental principle in geometric design: a global property of a curve often dictates its local properties. Here, the chirality of the spiral is a global characteristic that determines the sign of its curvature at every point. For two such curves to be joined with C^2 smoothness, their local properties (position, tangent, and curvature) must align perfectly. The constraint on curvature thereby enforces a constraint on the global properties, meaning only spirals that "turn" the same way can be smoothly connected.

Remark 2.3. This illustrates a general principle: local smoothness does not guarantee global geometric continuity. While it is always possible to construct a C^2 (or even C^∞) join between two curves at a point by adjusting derivatives, the global topological data—such as the cumulative winding number or chirality—may not extend through the join without interruption. In geometric computation, this corresponds to the loss or resetting of "phase information" across a smooth path.

2.4 FVISA *Plus Ultra*

To complete the construction, 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. One helix is defined by the original parametric equations, and the second by a reflected version of those equations.

Definition 2.4 (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 of all geometric computable manifolds (GCMs). The reference hyperplane is the xy-plane at $z = 0$. All geometric, group, and logical actions are measured relative to this frame.

Definition 2.5 (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:

Upwards-oriented: $z(u)$ is strictly increasing from base to apex.
Downwards-oriented: $z(u)$ is strictly decreasing from base to apex.

Let γ_A and γ_B be parametric curves defining Helix A (H_A) and Helix B (H_B), respectively. The helix H_B is defined as a rotation of H_A by 180° (or π radians) about the x-axis. This transformation maps a point (x, y, z) to $(x, -y, -z)$.

Let $\theta \in (-\infty, 0]$ and define $\theta = -\phi$ with $\phi \in [0, \infty)$. Then:

$$\gamma_A(-\phi) = (r\cos\phi,\, r\sin\phi,\, -h\phi), \quad \theta \in (-\infty, 0]$$
$$\gamma_B(-\phi) = (r\cos\phi,\, -r\sin\phi,\, h\phi), \quad \theta \in [0, \infty)$$

To apply the topological gluing, let H_A and H_B be compact 3-manifolds with boundary embedded in $\mathbb{R}^3$, representing the tapering helices. 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 conic cross-sections, each B_i is homeomorphic to the closed 2-disk D^2.

To construct the twin helical manifold, denoted by H_{AB}, we identify $B_1 \subset \partial H_A$ and $B_2 \subset \partial H_B$ via a diffeomorphism. Assume compatible parameterizations of B_1 and B_2, for example using polar coordinates (ρ, θ) induced from local cylindrical coordinates at the helix bases. A canonical identification map is

$$f\colon (\rho, \theta) \mapsto (\rho, \theta)$$

(i.e., the pointwise identity).

Let H_A and H_B be parameterized by smooth curves $\gamma_A : [0, 1] \to H_A$ and $\gamma_B : [0, 1] \to H_B$, representing the helical axes. To ensure C^1-smoothness at the join, we insert a short transition (spline) region of width $\varepsilon > 0$ near the base disk D^2.

Let $\mathbf{p}_A = \gamma_A(1)$ and $\mathbf{v}_A = \gamma_A'(1)$ denote the endpoint and tangent of H_A at the join, and $\mathbf{p}_B = \gamma_B(0)$, $\mathbf{v}_B = \gamma_B'(0)$ the corresponding data for H_B. Define the transition curve $S : [0, 1] \to \mathbb{R}^3$ as a cubic Hermite spline:[1]

$$S(s) = (2s^3 - 3s^2 + 1)\,\mathbf{p}_A + (s^3 - 2s^2 + s)\,\varepsilon\mathbf{v}_A + (-2s^3 + 3s^2)\,\mathbf{p}_B + (s^3 - s^2)\,\varepsilon\mathbf{v}_B$$

for $s \in [0, 1]$.

The full axis curve $\widetilde{\gamma} : [0, 2 + \varepsilon] \to \mathbb{R}^3$ is then given by

$$\widetilde{\gamma}(t) = \begin{cases} \gamma_A(t), & t \in [0, 1] \\ S\left(\frac{t-1}{\varepsilon}\right), & t \in [1, 1 + \varepsilon] \\ \gamma_B(t - 1 - \varepsilon), & t \in [1 + \varepsilon, 2 + \varepsilon] \end{cases}$$

The tubular neighborhoods about $\widetilde{\gamma}$ (with appropriate cross-sectional radii) define the smooth body of H_{AB}.

Now, as before, form the disjoint union $H_A \sqcup H_B$ and define the equivalence relation $\sim$ by $p \sim f(p)$ for all $p \in B_1$. The topological quotient space is

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

now, the embedding in $\mathbb{R}^3$ is globally C^1 at the join, with matching tangents and continuous geometry across the glued disk.

Geometrically, this construction realizes H_{AB} as a smooth union of the two helices, attached along a common base disk, with a transition region that interpolates smoothly between their axes, ensuring the entire manifold has at least C^1 (and, with higher-degree splines, potentially C^∞) regularity.

We term this new construction **Kaminsky Twin Helix** ("KTH"), or equivalently a *Twin Helical Manifold*. It is defined over the domain $(-\infty, \infty)$ with respect to the z-axis in $\mathbb{R}^3$, presenting not one but two escape vectors to infinity—within a single finite volume object.

Definition 2.6 (**Kaminsky Twin Helix and Unit Helices**)**.** A KTH is defined as a helical manifold in $\mathbb{R}^n$ formed by a topological gluing of two helical solids via a diffeomorphism at their respective bases. The two helices composing a single KTH are termed **unit helices**, denoted H_A and H_B.

Remark 2.4. Such a smooth manifold cannot be constructed by a simple "mirror and glue" operation on a surface of revolution like Gabriel's Horn, due to the lack of a well-defined tangent plane at the asymptote. The horn's rotational symmetry becomes a defect: overly rigid, it prevents the definition of a smooth gluing without singularity. Helical symmetry, by contrast, allows us to bypass this limitation and construct a FVISA object with unrestricted, doubly infinite domain. This controlled asymmetry will prove essential to the novel properties of the KTH.

Frustrated by the reemergence of the horn—now in doubled form—the topology demon retreats back to its abstract plane of being. Thus, with the help of an unlikely geometric ally, the day is saved. With the pedagogical crisis resolved, we may now examine the deeper structure of this manifold.

To return to the finite, we adjust each helix so that it no longer tapers asymptotically. Let $q > 1$ in $\rho(\theta)$ for both helices from this point onward.

By combining different chiralities and orientations, we define a geometric family of four canonical manifolds, denoted KTH_4. This family exhibits the following features:

- Smoothness of all orders: C^0, C^1, and C^∞
- Geometric representation isomorphic to the Klein four-group V_4
- Higher-order generalizations admit non-abelian symmetry groups
- Turing completeness under certain geometric encoding schemes
- Classical simulation of quantum-like computation via geometric flow
- Deep and unexpected links to the Yang–Mills existence gap and quantum formalism

2.5 The Group Structure and Geometric Topology of Twin Helices

Let us continue with the construction of a family of geometric objects working now in $\mathbb{R}^4$ composed of two unit helices to form 4 unique variations of the KTH. Our two variables of construction for unit helices are Orientation and Chirality as defined previously.

This yields a total 4 distinct invariant configuration states

Table 1: Essential Type Chart for Twin Helical Manifolds

Labeled	Chirality	Orientation	Axiality	Mnemonic
Type 1	Opposite	Opposite	Coaxial	Mirrored Horn
Type 2	Opposite	Same	Coaxial	Twisting Pair/DNA-like
Type 3	Same	Opposite	Offset	Mirrored Symmetrical Flow
Type 4	Same	Same	Offset	Pair of Horns/Drills

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 2.3 (Twin Helix Intersection Lemma)**.** *No twin helical manifold 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:

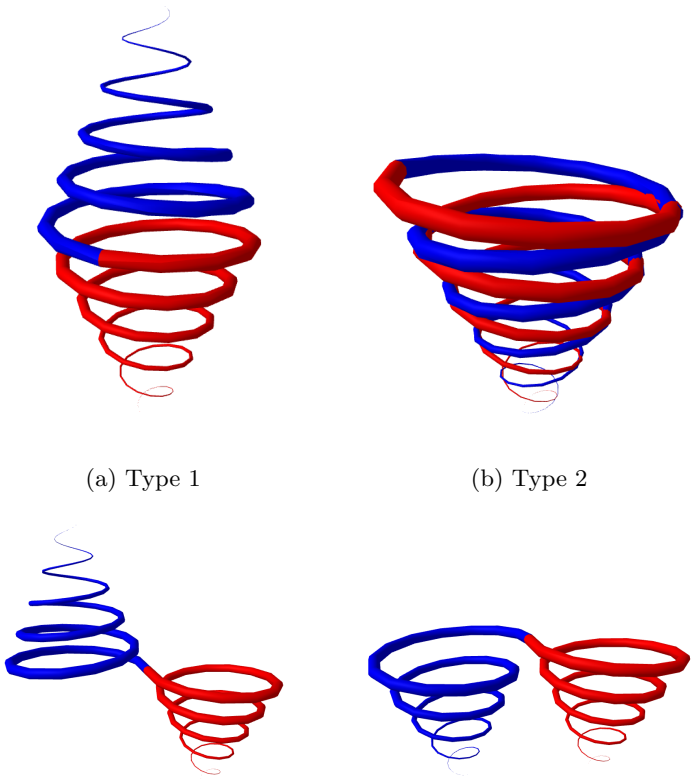
$$\alpha_x(t) = 2\pi N \chi_x t + \phi_x,$$

where:

N is the number of turns in one unit helix,
χ_x is the chirality of H_x, with $\chi_A = -1$ and $\chi_B = 1$ for Type 2,
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:

$$\alpha_B(t) - \alpha_A(t) = 2k\pi \quad \text{for some integer } k.$$



(a) Type 1 (b) Type 2

(c) Type 3 (d) Type 4

Figure 3: The KTH_4 Geometric Family.

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 2.5. 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.

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.

Proposition 2.4 ($KTH_4 \cong V_4$)**.** *The set of twin helical manifolds of type KTH_4 admits a group structure under the action of discrete transformations of chirality and orientation. This group 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,

$$KTH_4 \cong G \cong V_4.$$

□

Table 2: Group Progression of the KTH_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 KTH_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 KTH_{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

The following table of the calculated FSG progressions via the GAP has been provided for ease of reference.

In addition, the generators of KTH_{256} and lower orders are listed below.

As shown in Table 3, the group KTH_4 generated by these symmetries exhibits 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).

I 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 KTH_4, including its precise order are left open for further investigation, ideally by those with more expertise in computational geometric group theory. A more thorough enumeration of the group's elements and the relationships among generators would be an interesting direction for future work.

We will now explore the topology of the KTH. A single helical solid is best thought of as a compact 3-manifold with a boundary in $\mathbb{R}^3$ that also contains a single non-smooth boundary (the base). When we glue together the bases of two such unit helices in a proper manner, we form a larger compact 3-dimensional manifold with single boundary. The original bases become an internal surface, and the boundary of the new KTH consists of the original helical side surfaces and the tapered tips. KTH is of course homeomorphic to D^3 in topological terms, giving genus 0.

Singularities at the apex are indeed a threat, if the volume of the helix was indeed 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 or diffeomorphisms that produce smooth apex to KTH's unit helices.

Remark 2.6. KTH is defined as the topological union of two helical solids joined along their bases. This construction may introduce a non-smooth seam at the join, but we can consider this singularity idealized and computationally negligible, as the manifold is not used to model physical flow under Navier-Stokes constraints. We intend to compute under symbolic or topological conditions with idealized continuity assumptions. Those with more experience in topological surgery than the author are encouraged to provide a method for smoothing such a feature if it causes an issues should they require a perfectly smooth KTH interior in their own computations.

Proposition 2.5. *KTH is differentiable, continuous, and smooth everywhere (C^0, C^1, C^∞)*

Proof. KTH is constructed as the union of two smooth (C^∞) manifolds, each with boundary, glued together along their bases by a C^∞ diffeomorphism. By the standard compatibility of smooth structures under such gluings, the resulting manifold is C^∞ (and thus C^1, C^0) everywhere. Any non-smoothness at the seam can be made arbitrarily small (or eliminated) by choosing a sufficiently smooth gluing map and, if necessary, smoothing the cross-section function near the join. □

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

We now begin the construction of a Turing Machine ("TM") from a lattice of KTH manifolds. The feasibility of this approach rests on the fact that KTH, unlike most manifolds, provides a *rich but controlled* symmetry space. This is made possible by the independent adjustability of chirality and orientation of the unit helices.

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 KTH construction breaks this impasse.

3.1 Geometrically Computable Manifolds

Definition 3.1 (Geometrically Computable Manifold)**.** A **Geometrically Computable Manifold** ("GCM") is a smooth manifold composed of a finite arrangement of geometric substructures, each with finite-dimensional internal state. It is equipped with a locally defined set of transition rules such that the evolution of these states over discrete steps simulates any finite automaton—and, when sufficiently composed, simulates a Turing Machine.

More formally, a GCM is a 3-tuple $\mathcal{G} = (M, \Sigma, \Phi)$, where:

M A smooth manifold or structured geometric space
Σ A map assigning encodable states to substructures
$\Phi : (p, s) \mapsto (f(p), \tau(s))$ A local state transition rule composed of a diffeomorphism and state update

We elaborate each component in turn:

M: The manifold must be smooth, with well-defined local transitions and no bifurcation ambiguities. State evolution must exhibit finite branching at every encoding site. The manifold may carry a clock parameter t (e.g., tape-traversal or arc-length) to encode sequentiality. Globally, it must be piecewise-constructible from a finite set of geometric or topological primitives.
Σ: The state map

$$\Sigma : \mathbb{N} \to M$$

assigns discrete states to geometric locations (points, submanifolds, fibers, etc.). It is either partially recursive or totally recursive (if totality is specified externally). This map functions like a memory encoding or addressing system across the geometry.
Φ: The local transition rule is defined as a map

$$\Phi : (p, s) \mapsto (f(p), \tau(s)),$$

where $f : U \to V \subset M$ is a local diffeomorphism, and τ is a state transition function. The updated state is given by:

$$\Sigma'_V = \Sigma_U \circ f^{-1}.$$

Where:

$p \in U \subset M$ is a point or neighborhood in the manifold
$s \in S$ is the local state at p
f is a geometric transformation
τ is a state evolution function, either identity or nontrivial depending on f

Remark 3.1. So what is a GCM, really? A GCM is a manifold that can encode its own computable properties. A single KTH can qualify as a GCM. A lattice of KTH is also a GCM—just with more computing capacity. Even a braided, knot-theoretic, topological computer could count, if one could define suitable Σ and Φ over it. That said, the required asymmetries make such encodings difficult in practice.

3.2 Geobits—Bits, Geometric Bits, and State Space

Definition 3.2 (Geometric Bits)**.** A **geometric bit** ("geobit") is an element $s \in \mathbf{S}_{\mathrm{GCM}}$, where $\mathbf{S}_{\mathrm{GCM}}$ is a finite set of equivalence classes of embeddings of computable manifolds in $\mathbb{R}^n$, modulo ambient isotopy and global symmetry.

There exists a group G acting transitively on $\mathbf{S}_{\mathrm{GCM}}$, such that the pair $(\mathbf{S}_{\mathrm{GCM}}, G)$ defines a discrete geometric state space. The group action is isomorphic (as a group action structure) to a subgroup of the rotation group $SO(n)$.

Let $\mathcal{G}$ be a GCM that supports discrete encoding of state via symmetry-constrained transformations. A geobit is the minimal encodable unit of state in $\mathcal{G}$. It may be represented as an n-tuple:

$$s := (\chi, \omega, p) \in X \times \Omega \times P,$$

where:

X is the **chirality space**, typically $\{L, R\}$
Ω is the **orientation space**, typically $\{U, D\}$
P is an additional parameter space. This category is left intentionally open to accommodate future encodings within $\mathbf{S}_{\mathrm{GCM}}$.

These parameters together define a group structure—such as $V_4 \cong \mathbb{Z}_2 \times \mathbb{Z}_2$—when the composite operation $\chi{\cdot}\omega$ yields a closed, invertible group action under symmetry composition. All such operations are assumed to be deterministic and belong to a finite automorphism group acting on the embedding of the manifold in Euclidean space.

The discrete-time evolution of a geobit is governed by a deterministic group action:

$$\delta : s_t \mapsto s_{t+1} = g \cdot s_t,$$

where:

$g \in \mathrm{Aut}(\mathcal{G}) \subseteq SO(n)$ is a symmetry group generator
δ is a computable, deterministic state transition function.

Now, let:[1]

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

where each H_n is a configuration from the KTH_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 a unit helices are the same, assign bit value 1, if they differ, assign 0. This yields a total of 2 bits per KTH_4 configuration.

Theorem 3.1 (Geobit Capacity Theorem)**.** *Let KTH_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:*

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

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

Proof. 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: - In KTH_{16} (governed by $\mathbb{Z}_2^4$), each helix can encode 4 bits total: 2 bits from chirality and 2 from orientation. - In KTH_4, this reduces to 2 bits, since we encode only global relationships

[1] Magic happens here.

(i.e., quotient by identity swapping and pairing symmetries). - In KTH_2, assuming only orientation is swappable, we drop to 1 bit per configuration.

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

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

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 3.2. 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

Table 4 provides a reference encoding the relationships. Embedding will be explained later.

Remark 3.3. Given that there are 2^4 unique internal configurations of each KTH 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 KTH_4, which runs naturally in quaternary due to its $\mathbb{Z}_2 \times \mathbb{Z}_2$ structure.

It is important to note that *external point symmetries* (such as 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 3.3 (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 is especially useful when working with layered geometries or hybrid encoding schemes across different GCM classes.

The table 5 succinctly compares geobits with qubits via conceptual relations. [2]

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 and rotations 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

Regarding the construction of a "Hidden Variable State System". This terms demands some explanation. Recall that the superposition of a qubit is written as 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 the what is known in physics as a *hidden variable system.*

Definition 3.4 (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 to not 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 3.5. 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:

α is the classically accessible bit configuration, potentially masked
β he hidden morphism or transition operator responsible for the current or next state
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.

[2] Recall Holevo's Theorem.

Example 3.1. 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.

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

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 Down. Then we could write out the updated 4-geobit as

$$|1100|\beta\rangle_{C_2^4}$$

Since we know the full state the X's disappear. Though under V_4 this would quotiented down to

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

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 like such

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

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.

3.3 FSM Encoding for KTH

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

S A finite set of states
Σ An input alphabet
$\delta : S \times \Sigma \to S$ A transition function
s_0 An initial state
F A set of final (accepting) states or outputs

Proposition 3.2 (FSM Geometric Embedding Theorem)**.** *In 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:

$S \cong X \times \Omega$ — 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.)
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.

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.
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 a "vacuum" or "reset" configuration, and, if using quantum-inspired notation, by an HV-ket such as $|\alpha|\beta\rangle_G$, where α is the chirality configuration and β the orientation.
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.

□

The following chart provides a convenient reference.

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

Proof. Recall that

$$KTH_4 \cong V_4 \cong C_2 \times C_2$$

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 KTH_4 naturally carries two cyclic groups, its state space is sufficiently rich for this construction.

Define the state space for each KTH as a 2-geobit, encoding 2 classical bits of information.

Let L be a lattice of size $J \times K$, for $J, K \in \mathbb{N}$. Each cell in L contains a KTH_4 manifold (with unit helices A and B), which communicates symbolically with its adjacent cells.

Pixelated Screen: Suppose there is a readout layer P, with one pixel per KTH. Each pixel P_i displays as black or white based on chirality agreement:

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

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

1. Each cell H_i evaluates the orientation states of its neighbors H_{i-1}, H_i, H_{i+1}.
2. Applies the local update rule to determine the new bit b_i^{t+1}.
3. Updates its Helix A orientation and/or chirality accordingly.

Transition rules for Rule 110 use the group generators as defined in Section 2.5.
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:

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

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 KTH_4 manifolds is Turing complete. □

3.4 The GCM Lattice

Definition 3.6 (Kaminsky/GCM Lattice)**.** A *Kaminsky Lattice* ("KL") or *GCM Lattice* is a computational structure consisting of a collection of m discrete GCMs arranged in a lattice (e.g., $\mathbb{Z}^d$ grid), such that the system has a total geobit capacity of mn geobits, where n is the base geobit capacity of each GCM. Communication between GCMs can occur via symbolic or quasi-fluidic interactions realized as deterministic geometric deformations or flow.

Let:

$\mathcal{G}$ the configuration of the GCM lattice (the entire structure)
$\mu : \mathcal{G} \to \Sigma^N$ a projection mapping from geometric state to observable symbolic state, with $N = mn$ total geobits (e.g., as displayed on a black/white screen)
$\Phi : \mathcal{G}_t \to \mathcal{G}_{t+1}$ the dynamical update rule, governed by group generators $\{c_1, c_2, d_1, d_2\}$ or groupoid morphisms $\{m_0, ..., m_x\}$
$n \in \mathbb{N} \cup \{\infty\}$ the geobit capacity per unit; m the number of GCMs; $N = mn$ total geobits

The evolution of observable state satisfies

$$\forall t, \quad \mu(\mathcal{G}_{t+1}) = \Phi(\mu(\mathcal{G}_t))$$

provided that $\mathcal{G}$ is sufficiently complex to sustain N-geobit symbolic or quasi-fluidic dynamics.

Remark 3.4. The architecture of geometric computability is decidedly a Non–Von Neumann one. A KL 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, but also spatially distributed. Parallels can be draw to neural fields or reaction-diffusion systems if one requires an outside example.

Example 3.2 (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 KTH 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 2-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, bit transitions are collective, not local.

Remark 3.5. **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 KL consisting of a single GCM in a 1×1 matrix that still achieves infinite-geobit capacity. See discussion of the measurement problem in section 5.10.

3.5 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 that it depends. Like we established, the group operations determines the number of bits a quantized GCM actually 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. The reader is encouraged to think about it from a topological perspective, we are simply taking one 2-manifold and transforming it into another, more quantized 2-manifold, but there is no change in genus, they remain homeomorphic.

I will now turn to a geometric source to create a state space visualization of the geobits of KTH_4. Polytopes can naturally be used to represent the state space of geobits. In this case the vertices of the polytope are the KTH states, and the edges are the group transformations.

Starting with $V4, D4$ 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. This is in line with my previous assertions on non-abelian bit spaces. 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. During the course of this work it became apparent that a known, but not oft mentioned theorem causes a paradoxical result that prevents clean embedding of the KTH computational group model in a state space congruent to the dimensions that its geometry naturally lives in.

Proposition 3.4 (Minimal Embedding Dimension for KTH_{16} Symmetry Group)**.** *A faithful embedding of the group KTH_{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 KTH_4 naturally lives in 4D. However, the same limitation arises. In fact, a classic result [5] in the theory of compact Lie groups now asserts itself. *The maximal rank of an elementary abelian 2-subgroup that embeds in $SO(n)$ is $n-1$ (up to conjugacy).*

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 KTH progression, which becomes non-abelian above order 16 due to the use of semi-direct products.

Therefore, the state space embedding of KTH_{16} necessitates a "dimensional jump"—manifesting emergent 5D algebraic behavior, despite the geometric construction living in 4D. [3] □

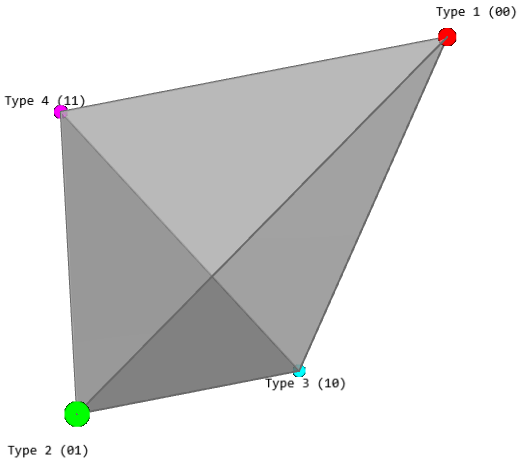
To embed all three original computational groups, we consider the full symmetry group of the 5-dimensional hypercube (the penteract), which is the Coxeter group B_5. This group has order $2^5 \cdot 5! = 32 \cdot 120 = 3840$. The relevant rotation subgroup, called the *demipenteract*, is $D_5 \cong (C_2^4) \rtimes S_5$, 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.

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

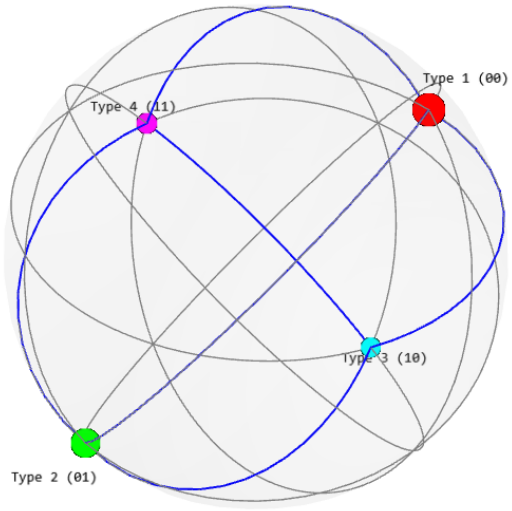
Since it is not possible to faithfully visualize a 5-dimensional shape, we settle for visualizations of V_4 and the regular tetrahedral group T (of order 12, embedded in $SO(3)$).

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 4a displays such a rendering, with colorings chosen to highlight these symmetries.

Remark 3.6. 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 4b illustrates this method. This discrete tessellation highlights the analog 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 4: 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 3.7. The tetrahedral structure of the V_4 GCM state space is 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 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.

3.6 The Geometry of Logic

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:

1. **Ternary / n-ary logic**: 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.

2. **Fuzzy logic**: 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

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

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:

1. Allowing arbitrary join angles breaks the natural algebraic group structure of KTH_4. Closure under group operations is lost when joins are not restricted to a finite set.

2. 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 3.8. 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 similar to Noether's Theorem of Conservation but for geometrical computational logic. 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 3.5 (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.

One path becomes a clean, algebraic infinite-state computation model. The other path devolves into an increasingly purely geometric analog, continuous processor. A KL made of KTH_4 sits at the cusp of this tradeoff. Its geometric expressiveness is actually quite high—as we will see. In classical physics, symmetry preserves energy and momentum, but Lemma 3.5 in GCT preserves computability and algebraic closure within a tightly constrained topological economy.

Remark 3.9. 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 we have only derived a single form of GCM, one can imagine higher order manifolds where can you deform the joint outright into asymmetric 1-manifolds. These higher order manifolds would devolve into pure fluidic computability, leaving logic altogether and entering increasingly abstract and chaotic forms of analog computation.

Example 3.3. 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 3.7 (Orientation Bit Assignment for n-ary Unit Helices)**.** Let $\mathbf{v}_i$ be the axis of symmetry of helix i, and let $\hat{z}$ be the reference axis. For n-valued logic, the orientation bit b_i is defined as

$$b_i = \left\lfloor \frac{n}{2\pi} \arccos\left(\frac{\langle \mathbf{v}_i, \hat{z} \rangle}{|\mathbf{v}_i|} \right) \right\rfloor, \qquad b_i \in \{0, 1, \ldots, n-1\}$$

This generalizes the original ternary mapping formula, allowing the angular state of a helix to directly encode n-ary logic. The intrinsic property of a helix—its capacity to twist through any given space—ensures that orientation is always well-defined via its internal symmetry axis.

The following table helps organize our models again showing the relationship between join symmetry and the resulting system that comes as a result of it.

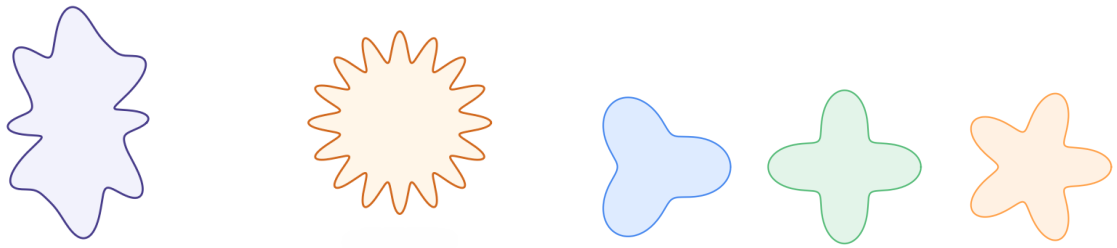
Geometric Expressiveness	**Join Symmetry**	**Algebraic Closure Group**	**Result**
Extreme	0-fold	Diff(M) (Zero)	Computational gauge field
High	1-fold (360°)	C_1 (Trivial)	Geometric analog logic flow
Medium	2-fold (180°)	C_2	Classic boolean logic gate
Low	4-fold (90°)	V_4	Quaternary logic gate
None	n-fold (Infinite)	$SO(2)/C_\infty$ (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. The analogy to Earth's own habitable zone in the solar system is intentional.

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. The physically stable and computationally tractable zone is inherently narrow.

For n-ary logic, the cross section of the KTH 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 5b.

However, one can readily imagine more expressive cross sections that realize the required angular structure but offer even richer forms of expressiveness. Figure 5a illustrates such alternatives and compares expressiveness across different logical regimes.



(a) Expressiveness in Binary vs Hexidemical

(b) n-logics in Cross Section Form—3,4,5-ary

Proposition 3.6. *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. Consider 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

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

which is a circle. Thus, under arbitrarily high-order rotational symmetry, the only possible closed 1-manifold in $\mathbb{R}^2$ is the circle. This follows from Lemma 3.5. □

Definition 3.8 (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.

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.

Remark 3.10. 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. Note 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 space governed by a compact simple Lie group.

Conjecture 3.7. *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) are suitable as GCMs.

Remark 3.11. 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 the purposes of classical computation in so far as the manifold remains smooth. This is a valid point, but 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.

Definition 3.9 (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 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).

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 turns bends at a smooth 90° angle. It is an inflection point between chiralities.

Definition 3.10. Chirality Inflection Surface. Any twin helical manifold made of opposing chirality unit helices can be achiral, even though the unit helices on either side of the glue joint are not. The twin helix contains two chiralities within a single manifold canceling each other out. The point at which $\chi = 0$ is termed the Chirality Inflection Surface.

Remark 3.12. The idea of a *chirality inflection surface* or point is a fascinating subject. These are by definition are endogenous to KTH Type 1 and 2. If fluid followed from the top of a KTH Type 1 to the bottom, it would start with a certain angular motion defined by the first chirality but eventually switch and exit with angular motion defined the opposite. I have provided a definition, but the richness cannot be fully explored given the constraints of this paper. As a soft proof sketch—let us imagine there exists a bijection β between all points on unit helices $H_A \leftrightarrow H_B$ such that every point is symmetric to some shared θ. Yet KTH is a smooth manifold, despite chiralities being scalar values. Then via Intermediate

Value there must be some point where $\chi = 0$ as you travel the full length along this bijection, having chosen two points. One at the apex of H_A that is traveling to the apex of H_B and vice versa. This surface happens at the glue joint of H_{AB}.

This explicit geometric sacrifice embodies Lemma 3.5 yet again. Greater logical expressiveness at the expense of geometric sharpness. The resulting manifold supports ternary logic in the chirality subspace, note that is not merely masked/unread ("X") values of the hidden variable state system.

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" [4] an inflection point between winding and unwinding. This asymmetry reflects the deeper topological distinction between angular (periodic) and chiral (binary with singularity) symmetries.

Remark 3.13. These changes however causes 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, 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 × 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 3.4. 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

$$|U014|(d_2)_B\rangle(\mathbb{Z}_{15})^2 \xrightarrow{d_2} |U015|e\rangle(\mathbb{Z}_{15})^2$$

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*.

Going further we would also do well to admit that a GCM is at its core, not necessarily constrained by purely symbolically derived bits 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. But there are more variables at play once vectors get involved.

3.7 Syntax, Semantics, and Computation: A New Synthesis

Remark 3.14. **Warning**. The remainder of this section ventures into metamathematical and existential territory, tracing the consequences for science, technology, and society at large.

It is at this point 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 and Amr Sabry'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, Tae-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.

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

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 8). 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 3.11 (Microstate)**.** Let $\mathcal{S}$ be the state space of a computational or geometric system (e.g., a configuration of a GCM, a set of logical variables, a geometric arrangement of bits, etc.).

A microstate is a single, completely specified, fine-grained configuration of the system—an assignment of all local, atomic variables and parameters that determine the system's instantaneous state with maximal precision. Examples:

- Geometric Computation: a microstate is the exact assignment of chirality, orientation, flow, etc. to every helix in the lattice.
- Programming Languages: a microstate is the full set of variable values, stack frames, program counter, etc., at a given step of execution.
- Thermodynamic System: a microstate is the exact position and velocity of every molecule.

Definition 3.12 (Macrostate)**.** A macrostate is an equivalence class of microstates, characterized by some coarse-grained, collective, or emergent property or observable of the system.

That is, many distinct microstates may correspond to the same macrostate if they agree on certain global features, summary statistics, or logical/physical invariants. Examples:

- Geometric Computation: a macrostate can be "all configurations with total chirality zero," or "all with the same winding number sum," or "all corresponding to the same output under a projection map μ."
- Programming Languages: a macrostate can be "all states where a certain function is about to be called," or "all stack traces with the same call graph."
- Thermodynamic System: a macrostate is defined by bulk properties like temperature, pressure, and volume. Each macrostate contains many microstates.

Mathematically: If $\mathcal{M}$ is the set of all microstates, and $\sim$ is an equivalence relation (e.g., "has the same projection under some observable"), then the set of macrostates is $\mathcal{M}/\sim$.

Definition 3.13 (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 3.14 (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 and Sabry'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). 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 3.15 (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 3.16 (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:

- 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-manageable (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 the ways in which meaning is constructed and communicated among its users.[5]

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, in order to maximize accessibility and highlight the new conceptual terrain. So with the parts in place, we now can *macro-express* a new metatheorem:

Conjecture 3.8 (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.

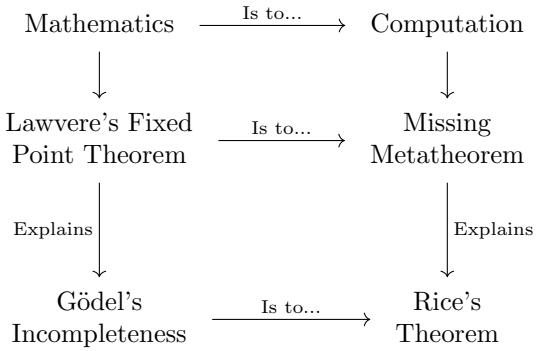
This is the central motivating result of this work.[6] It is the heart and soul of my philosophical approach to understanding the nature of computation. It simply generalizes Lemma 3.5 to all computation. I contend that there exists a deep connection here at work between verifiable ("algebraic") and expressive ("geometric") computational properties and that this applies to all deterministic computing systems. I 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 generalization 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 Godel's Incompleteness. Rice's Theorem states that "All nontrivial semantic properties of Turing machine languages are undecidable." Godel's incompleteness states "There will always be statements within a formal system that are true but

[5] If you are confused, I can assure you that you aren't alone. The author *thought* he was creating a new tool for computation—it turns out he *actually* ends up tripping over why groups of people have trouble creating *shared systems of meaning*.

[6] I understand Yang-Mills is also important, but as far humanities computational day-to-day is concerned—my apologizes to physicists of the world.

cannot be proven." Godel and Rice are in fact 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 obviously 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 of programming languages, means 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 3.6. 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 defeat Rice's Theorem, but the laws of physics and the constraints of engineering decrees this shall not be so.

But binary logic however 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. [7] 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,

[7] 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. Here, we can reverse the Church-Turing Thesis—the hardware actually refuses to compute anything besides the program.

uncovered this when he argued for macro-expressiveness as the defining metric of expression in his seminal 1991 paper on the subject [13].

The EVT cleanly explains the odd "dual nature" one sees in functional programming languages. One side offers strict typing with meta-theoretical certainly during runtime but but meta-linguistic restrictiveness at compile time. The other side with dynamic typing offers meta-linguistic expression in construction but no meta-theoretical certainty at runtime. Attempts have been made to bridge the gap from both sides (MetaOcaml is to Coquand, as Typed Racket is to Felleisen), but—use one too many features of these powerful languages and they can run into instability. **An explicitly stated conservation law for 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 the curve? There are many computational processes that are both low in expressiveness but also low in verifiability. An example being the infamous "Turing Tarpit" where everything at once 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 to must my own analogue to Lawvere's Fixed Points cover more conceptual ground when dealing with entire systems of logic. **My contention is that Lawvere's construction on its own does not have the power to capture the 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 than 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 with regards to the actual 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 why I use only a quasi-Noetherian framing with regards to 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.

This is why this digression must be included here. In GCT, modeling EVT relationships becomes much more tractable, since there is no separation between logic and memory. A KL is an idealized version of a Non-Von Neumann architecture existing on substrate defined purely by math. [8] Here one can *see* how the operational semantics forcibly deforms the computational substrate, and model how increasing algebraic closure eats away at the possible expressions in geometry with increasingly verifiable but near infinite amounts n-ary logic with every increasing bit.

Remark 3.15. Regarding tractability, hardware is not truly a concern for geometric computation. If one had a stacked array of KLs 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 in real life where 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, static boundaries between code and metacode. As a result, although I must state the entire trilemma as I have discovered it, I will not be exploring tractability as much in this paper, as it is more of a physical constraint.

As Baez writes in *Getting to the Bottom of Noether's Theorem* [8]

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

> "Atiyah [14] 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 own argument may be coincidental, it is nonetheless unsettling. I do agree with 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 cannot not be cheated. Under the watchful eye of the ZFC, algebra and geometry can play nicely together, but the act of computation creates a bottleneck. Though, I must acknowledge that Atiyah himself likely never anticipated for his words to be used in this way.

Even so, I fully acknowledge my claims here border on being outright inflammatory—and go against multiple trends in current computer science, especially programming language research. Still, given the implications for humanity at large with our increasing dependence on software, I will say that the failure to understand the underlying metamathematics of this phenomenon and of computation at large on a categorical level is rather unfortunate. And I am not alone in this sentiment. As Joshua Meyers [15] writes:

> *From Turing machines all computations can be executed, just as from set theory all mathematics can be constructed. But as category theorists, we know that a particular set-theoretic implementation of a concept does not qualify as the essence of the concept, since it could just as easily be implemented infinitely many other ways. The true essence of a concept is its internal structure, and how it relates to other concepts (its "external structure", perhaps). Thus from the categorical viewpoint, Turing machines do not succeed in essentializing computation any more than von Neumann ordinals essentialize the naturals.*

[9]

Remark 3.16. 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 of any consolation, the author shared such feelings when writing this.

The convergence of categorical natural language theory and the abstractions of higher category and ∞-topos theory signals an emerging synthesis. That this validation of ur-metatheory should depend on insights from a domain historically neglected—natural language, meaning, and learning—is a poetic turn, and 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 however hints that the latent computational power within geometric objects has yet to be fully unlocked. Let's turn the key.

[9] For those readers not familiar with the philosophical background here, see [16].

4 A Kleene-Theoretic Approach to Foundations for GCT

Before proceeding further, I will rigorously establish GCT in the language of Kleene's theory of partial computable functions.[10] Kleene's formalism is particularly well-suited here, as it offers a foundational framework for modeling computable processes that is platform-independent and does not rely on the specific machinery of the lambda calculus. Moreover, the inclusion of partial functions and groupoids fits naturally into this setting and prepares the groundwork for the following section.

Stephen Kleene's pioneering work in the 1930s and 1940s established the modern theory of computability, most notably through the definition of partial recursive functions. These form the mathematical backbone of the classical model of computation, providing the foundations for Turing machines and the Church-Turing thesis.

4.1 Fibered Kets and Tensor Products

In analogy with the HV-ket, we now introduce a "fibered ket" notation appropriate for geobits, as we move beyond purely symbolic computation.

Definition 4.1 (Fibered Ket Notation for Geobits)**.** A **fibered ket** is an element of the tensor product space associated to a geobit $g \in \mathbf{S}_{\mathrm{GCM}}$:

$$|\alpha\rangle \otimes |\beta\rangle \in \mathcal{H}_{\mathcal{G}}$$

where:

$|\alpha\rangle$ is the classical projection—encoding observable chirality and orientation,
$|\beta\rangle$ is the internal (hidden) state, which may include fluidic variables, topological twists, or other degrees of freedom,
$\mathcal{H}_{\mathcal{G}}$ is a Hilbert-like geometric phase space in which the tensor lives, parameterized by the groupoid $\mathcal{G}$ (for discussion of parametric groupoids, see Section 5.6).

The evolution of geobits is defined as a function on the tensor product space:

$$(|\alpha\rangle \otimes |\beta\rangle) \mapsto |\alpha'\rangle \otimes |\beta'\rangle.$$

The fibered ket encodes both the observable and hidden structure of a geobit, and its manipulation is analogous to taking products of vector fields. While the full interpretation of this formalism will be developed in the following section, from this point onward we will treat a GCM as an object that can be subjected to computational "flows"—idealized incompressible fluids represented by finite vector fields. As this section unfolds, I will demonstrate how this notation fits naturally into the composition and recursion schemes familiar from Kleene's theory of partial recursive functions.

4.2 Partial Recursive Functions—Primitives and Composition

The partial recursive functions are built from three primitives:

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

And three closures:

Primitive composition - $h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_k(\vec{x}))$ combines any of the above three functions
Primitive recursion - Given $f(x)$ and $g(n, y, x)$, define: $h(0, x) = f(x), \quad h(n{+}1, x) = g(n, h(n, x), x)$
Unbounded μ-recursion - Define $f(x) = \mu y[g(x, y) = 0]$ with the smallest y being that which makes the function vanish. This allows search and potential non-halting in a computational process.

[10] I realize my order of exposition is unconventional; I chose to begin constructively, as it offers more intuitive grounding for the reader.

These 6 operations define partial recursive functions everything computable by them is what we now call computable in the classical sense. Let $\mathcal{G}_f$ denote the class of functions computable by a GCM. I now construct a geometric interpretation of each PRF constructor. The primitives are trivial to represent.

The **Zero function** is congruent to identity of the group, starting from an unread state of geobit g in $\mathbf{S}_{GCM}$:

$$Z(x) = 0 \cong |X^{I(G)}|e\rangle_G \xrightarrow{e} X^{I(G)}|e\rangle_G$$

Or equally multiplication by the identity matrix of the vector field β

$$|\alpha\rangle \otimes |I\rangle = |\alpha|e\rangle$$

The **Successor function** is congruent to any substructure operation that manipulates one tuple of geobit space:

$$S(x) = x_i \cong |\alpha_0|\beta\rangle_G \xrightarrow{\beta} |\alpha_1|e\rangle_G$$

Where $\alpha \in (X, \Omega, P) \subset g$ where P is any other element subject to quantization. [11]

The **Projection function** is congruent to:

$$P_k^n(x_1, \ldots, x_n) = x_k \cong \text{Readout}(P_i)$$

or any readout similar to the readout function from 3.3 that reads part of the overall tuple in g. Geobits are discrete objects so this is taken as a given. While normally this would be the orientation bits as they are the densest, readout can be defined as whatever makes sense for the computation that is being modeled—as we will see helices can have rich geometric states.

Primitive Composition is defined as composite flow of transformations. There is an applied transformation to the fiber $|\alpha\rangle \otimes |\beta\rangle$ via a hidden variable, say flow rate:

$$|\beta\rangle \mapsto |v(\beta)\rangle$$

Then, the geobit is updated via a function of the fiber:

$$|\alpha\rangle \mapsto |s(\alpha, v(\beta))\rangle$$

Where: $v(\cdot)$: transforms the flow rate $s(\cdot)$: acts on the chirality, but depends on the output of v such that:

$$v(x, y) = \begin{cases} +1 & \text{if } y > 0 \\ -1 & \text{if } y \leq 0 \end{cases}$$

where $y = v(\beta)$.

Example 4.1. We define the initial chirality $|\alpha\rangle = |+1\rangle$ and initial flow $|\beta\rangle = |0.7\rangle$ so the total state is: $|+1\rangle \otimes |0.7\rangle$

To apply the hidden transformation we can define $v(\beta) = \beta - 1$, as this is a stepped function that just "slows down" at each step. If we apply v to the flow variable:

$$|\alpha\rangle \otimes |v(\beta)\rangle = |+1\rangle \otimes |-0.3\rangle$$

To apply an update to the observable state, controlled by the new flow we use s to update chirality based on the new flow rate:

$$s(\alpha, v(\beta)) = \begin{cases} +1 & \text{if } v(\beta) > 0 \\ -1 & \text{if } v(\beta) \leq 0 \end{cases}$$

Here, $v(\beta) = -0.3 \leq 0$, so the new chirality is -1. The updated state: $|-1\rangle \otimes |-0.3\rangle$

Behind the scenes, the explicit matrix calculation when using tensor products works out to:
$|\alpha\rangle = (1, 0)$ ("left-handed")
$|\beta\rangle = (0, 1, 0)$ ("medium" flow)
Tensor product: $|\alpha\rangle \otimes |\beta\rangle = (1, 0) \otimes (0, 1, 0) = (0, 1, 0, 0, 0, 0)$

[11] Again, I don't know if there are any other manifolds out there, I'm trying to keep definitions open.

Definition cases for **Recursive Composition** are defined as such. Let: $G_0 = F(\vec{x})$ be the initial geometric configuration and let $G_{n+1} = G_n \triangleright T_n$ be the recursive case Where "right action" ($\triangleright$) matches the application of the next "move" to the current state, preserving the order of operations. It applies a local transformation T_n, be it a twist, swap, or evolution, to the previous state. Each T_n is an automorphism, an invertible transformation of the the manifold onto itself that represents a single step of the geometric recursive process.

To demonstrate this state in fibered ket notation, let the state at step n be:

$$|G_n\rangle = |\alpha_n\rangle \otimes |\beta_n\rangle$$

Where $|\alpha_n\rangle$ is the observable and $|\beta_n\rangle$ is the internal variable.

The initialize we set:

$$|G_0\rangle = |F(\vec{x})\rangle = |\alpha_0\rangle \otimes |\beta_0\rangle$$

Where F is the geometric "constructor" it picks the initial state either at random or by evaluating an initial function or setting up initial twist/flow parameters.

Applying the recursive case means that each update is:

$$T_n = T_n(\alpha) \otimes T_n(\beta)$$

a local transformation that can act separately or jointly on observable and hidden variables

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

Or, for more general transformations, T_n could be a coupled operation.

After k steps the state is

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

This is congruent to basic recursion of the definition in PRF.

Example 4.2. Commutative operations with basic recursion. Suppose our initial state is:

$$|\Psi_0\rangle = |\text{left}\rangle \otimes |\text{up}\rangle$$

Chirality swap and Orientation flip automorphisms respectively are:

$$C \text{ (left} \leftrightarrow \text{right)}] := |\Psi_1\rangle = (C \otimes I)\,|\Psi_0\rangle$$
$$O \text{ (up} \leftrightarrow \text{down)}] := |\Psi_2\rangle = (I \otimes O)\,|\Psi_1\rangle$$

Composition becomes:

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

Follows into a final expanded form:

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

Notice, that for these automorphisms, the order doesn't matter as C and O act on different factors.

We interpret **μ-Recursion** as a geometric search process within the GCMs or lattice of GCMs. Let:

Geometric Search - Construct a family of geobit configurations $|G_y\rangle$, each representing a candidate value y, via sequential application of a group action or local automorphism.
Halting Condition - Be the process halts if, for some y, the configuration $|G_y\rangle$ satisfies a geometric criterion such as set of symmetries or special region.
Partiality - Exist if no such configuration is reached, the process continues indefinitely, and the result is undefined on that input.

Example 4.3. Infinite Loops in a minimal KL Suppose we have two neighboring GCMs (A and B), each with:

$|\alpha\rangle$ as orientation (up/down)
$|\beta\rangle$ as continuous flow rate

The full joint state then becomes:

$$|\Psi\rangle = (|\alpha_A\rangle \otimes |\beta_A\rangle) \otimes (|\alpha_B\rangle \otimes |\beta_B\rangle)$$

Similarly to example in primitive composition, but this time instead with orientation instead of chirality. Local transition rules are defined as:
GCM A: If β_A (flow rate) increases past a threshold of 1 (say $2 \to 3$), swap orientation α_A. After swapping orientation, send a signal to GCM B to increase β_B.
GCM B: If β_B increases past the same threshold, swap orientation α_B. After swapping orientation, send a signal to GCM A to increase β_A).

Thus forming an infinite loop. To express in fibered ket notation we define the transition as an operator T acting on the joint state:

$$T : |\Psi\rangle \mapsto |\Psi'\rangle$$

where T is the composition of "flow increase and orientation swap" in each GCM, triggered by the other's state.

Iterating T produces an infinite sequence:

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

with no halting configuration. One can assume the local halting condition of "flow rate equals zero" is never reached. The full system state evolves as:

$$|\Psi_{n+1}\rangle = T\,|\Psi_n\rangle$$

where T alternates flow-increase and orientation swap in A and B. If the triggering conditions never reach a halting configuration, the system loops endlessly, mirroring the classical search for the minimal y in PRFs. In this way, μ-recursion is realized as an unbounded sequence of fibered ket transitions with no fixed point.

4.3 Kleenes Theorems for Computational Groupoids

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 [17] states that fixed points are a fundamental requirement for computation to exist. Information may indeed exist without the use fixed points but computation cannot. Computation by definition requires distinction, and all distinction requires a fixed frame of reference. This merely a new and interesting way in which Kleene's theorem has manifested itself, albeit without the typical language of 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 4.1. *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 4.1. If we assume there is an state encoding then geometric system can encode its own configuration (group element, fibered ket state, parametric vector) as part of its own "internal data." Let's 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 own 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 4.2. *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 4.4. 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 4.2. 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 it is essential that there is the 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 doing so carefully with respect to the computability and the topology of any $\mathcal{M}$, once collaborative refinement of categories in $\mathcal{ESM}^+$ is formalized. I've presented here a strong analogy and partial construction, but leave a full formal proof for future work.

5 A Geometric Topology Framework for Discrete Fluidic Computation

Recent work by Miranda et al. [18] on the computability of fluid flows—especially the Turing completeness of Euler's equations as conjectured by Tao [19]—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 high school science experiment in which a hose filled with water is placed across a vibrating speaker. The resulting shape of the hose—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 [20] visually validate the emergence of smooth, helical manifolds in physical systems driven by periodic forcing. Such phenomena provide direct, tangible evidence that the logic of geometric computation and bordism is realized in nature, with chirality and smoothness as optimal solutions for dynamical storage and flow.

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

Remark 5.1. Recall Theorem 3.1. That result holds for a purely symbolic computational system. However, even with binary rotations between helices, the true bit space encodable within a single KTH under C_2^4 is likely on the order of 5–10 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")

Thus, the geobit capacity grows quickly as more independent physical parameters are included. While it is theoretically possible to simply 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 ultimately "read" from the overall geobit value. In fluidic (physical) systems, they may become inaccessible or non-local.

Remark 5.2. The vibrating hose experiment can be viewed as an analog of a *geometric bordism*, where the dynamic shape of 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 hose. See section 6.5 for a breakdown on cobordism theories.

Proposition 5.1. *The geometric topology of a GCM encodes its own information-theoretic structure, but it is also is dependent on the subjective quantization of the model. Geometric computation is deterministic, but relativistic.*

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, quasi-fluidic, fluidic, and gauge regimes is thus not an objective fact about the system, but a reflection of how *we choose* to read and utilize its structure a computational relativity at the heart of the theory. When the shape and flow are the substrate, the encoding choice defines computing model, and geometric deformation lets one interpolate between discrete *and* continuous logic.

This doesn't mean that we will not give a definition of 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 a explicitly defined information stream to drive behavior, even a hidden one, as opposed to just an explicitly defined algebraic operation is the threshold that has been crossed.

5.1 Fluidic State Systems

We will now introduce the concept of a *Fluidic State System* ("FSS"). Semantically, this is done to show the non-mechanistic nature of GCMs [12] Classical FSMs cannot represent computations that depend on continuous parameters, boundary behaviors, or topological bifurcations, things that are key 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 a 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.

Definition 5.1. Fluidic State System. A FSS is a dynamical system defined by a smooth manifold M equipped with a finite or countable set of distinguished regions $\mathcal{R}_i$, a family of continuous flows $\{\Phi_t\}$ on M 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:

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

The FSS variables are listed below with explanation:

States - are the configurations of helices $\{(\chi_i, \omega_i, \tau_i, r_i)\}_i$ corresponding to symbolic states
Inputs - are vector fields $\phi(t, x)$ across space
Transitions - diffeomorphisms in the GCM are governed by geometric differential equations or group and groupoid morphisms
Outputs - are stable geometries or emission of secondary flow patterns

The conceptual map in Table 6 is provided for ease of the reader.

Table 6: Classical vs. Quasi-Fluidic Computation

Classical Computation	**Geometric Computation - Quasi-Fluidic Mode**
Finite State Machine (FSM)	Fluidic State System (FSS)
Turing Machine (TM)	Kaminsky Lattice (KL)
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 $\mathcal{R}_i$

Remark 5.3 (Comparing FSS Definitions). The core definition of a Fluidic State System (FSS) given above as a 4-tuple $(M, \{\mathcal{R}_i\}, \{\Phi_t\}, \mathcal{I})$ emphasizes its geometric and dynamical character, suitable for most applications in geometric computability and fluidic computation.

[12] And yes, to also avoid any future acronym namespace collisions with "FSM".

For readers familiar with automata theory, an alternative and fully equivalent formalism is available using a 6-tuple:

$$\mathcal{F} = (M, \Phi, \mathcal{G}, \Delta, \mu, \rho)$$

Here, M is the local state space, Φ is the family of local flow or routing rules, $\mathcal{G}$ is the group or groupoid of allowed geometric transformations, Δ is the evolution or update function; μ is the observation or readout map, and ρ encodes the clocking or timing regime.

The two approaches are interchangeable in practice, we use the 4-tuple here since it is ideal for intuition, the 6-tuple for categorical or algorithmic analysis.

[13]

Theorem 5.2. *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 M ($|\alpha\rangle \otimes |\beta\rangle$), equipped with a groupoid $\mathcal{G}$ of geometric transformations (twists, flows, orientation rotations).

Local evolution is governed by continuous flows $\{\Phi_t\}$ 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. □

5.2 Composition of Quasi-Fluidic GCMs in a KL

Previously we assumed purely symbolic communication between lattices of GCMs, forming a KL. 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 topological computation a more fitting model might be to not model GCMs as discrete units at all. Rather they are 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.

Hence, the potential path to 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 just 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. Consider 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 1 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 2D KL. The U state of "No Twist" ("NT") represents "refusal"—neither input nor output. Hence one could write code such as:

[13] Feedback here is welcome, I am open to combining these two together...maybe. I will leave this footnote for a few weeks after the preprint.

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

The 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 a field programmable gate array (oft abbr. "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 KTHs in a particular row to into a "halt" state. The compiler would automatically generate machine code that would identify all even numbered KTHs 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:

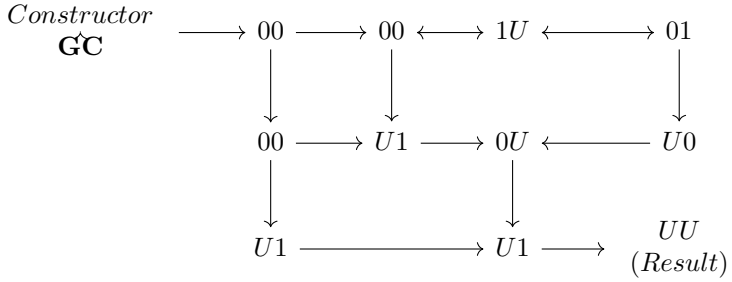
4 single-directional flows: {U}, {D}, {L}, {R}
6 two-directional flows: {U, D}, {U, L}, {U, R}, {D, L}, {D, R}, {L, R}
1 null direction: {}

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 computation 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 KL 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.

5.3 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 6.1. 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 both Dr. Tao and Miranda have 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 was 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 KL, the problem and the computer can *become one and the same.*

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

Continuity The vector field $\vec{F}(x)$ must have no sudden jumps or discontinuities.
Differentiability $\vec{F}(x)$ must be at least C^1.
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.
Boundedness or Decay at Infinity $\|\vec{F}(\vec{x})\| \to 0$ as $\|\vec{x}\| \to \infty$.
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 smooth flow of fields over smooth manifolds.

5.4 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 KL. It is not my 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 which serves to deepen our understanding of quasi-fluidic computation.

Definition 5.2 (The Fluidic Helix Computational Model ("FHCM"))**.** The FHCM is an example of a dynamical system that, when sufficiently composed, can simulate computation of helices in a KL. 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:

Chirality χ**:** $\in \{+1, 0, -1\}$—right-handed, left-handed, or undefined. Governs direction of flow and computation.
Orientation ω**:** $\in \left\{\frac{2\pi k}{N} : k = 0, \ldots, N-1\right\} \cong C_N$ (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.
Flow Rate $\phi(t)$**:** $\in \mathbb{R}$—fluid speed through the helix body. A generalized hidden variable that can drive state changes. Quantized at the user's discretion.
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:

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."
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.
Tube Volume $V(t)$**:** —can be seen as analogous to data throughput, or more aptly, parallelism. Greater volume allows for more flows.
Phase Offset $\theta_{\min}$**:** $\in [0, 2\pi)$—the starting point in the twist cycle. Can serve as an "initial state" or "addressing" mechanism.
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 as they see fit: 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 a global pulse causing a local change in $\phi(t)$ as dictated by each individual GCM transmitted across a KL. 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 5.9).

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 modular logic gates. The model is thus highly flexible: it recovers classical digital logic as a special case, while supporting richer logics for advanced forms of computation. (The examples below are presented under binary ω-logic.)

AND/OR gates — Both input KTH have their chirality set to route into the downstream KTH, whose internal transition rules encode the relevant truth tables for the operations. A flow pulse then executes the operation.
NOT gate — One input KTH routes to a downstream KTH, and it automatically inverts the given orientation on a flow pulse.
Memory register — Configuration is stable as long as chirality and orientation remain unchanged and no flow pulse disrupts it.
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 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 why we use the term "modular logic gate".

Definition 5.3 (Modular Logic Gate)**.** Let $N \in \mathbb{N}$ with $N \geq 2$. A *modular 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 modular gate. For $N \to \infty$, f is an analog gate over S^1.

Remark 5.4. 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 modular 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 5.1 (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 [21] 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 5.5. I must stress that these are merely interpretive tools,[14] 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.

5.5 The Algebra of Flow

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 3.6, reappears. The relevant vector space now represents a field flowing through a bounded geometry.

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 *real-valued macro-expressiveness* in computational logic. It quantifies the total geometric variation in a cross-section compared to an isotropic flow volume.

[14] There are a clever ways of saying when one is a bit too far into the weeds here.

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 [13] 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 3.6, binary logic is the most expressive form of discrete logic—less is more in this regime. 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 one and 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 5.6. 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 I do not dispute. In real systems, noise always intervenes, but this does not preclude a normalized measure of expressiveness itself. My contention is 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 5.3 (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:

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

In the absence of symmetry ($n = 1$), all $2K+1$ modes with $|k| \le K$ 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.

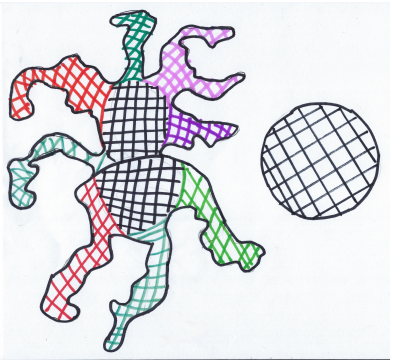


Figure 6: Macro-expressiveness vs Isotropy

Thus, increasing symmetry (micro-expressiveness) necessarily reduces the space of physically expressible patterns (macro-expressiveness). 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 a 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 ask "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 5.7. **Warning.** The EVT 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 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 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 [15], 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 not all forms of analog computation are created equal, in the same way that that Cantor showed there are more real numbers than natural numbers. Pour-El and Richards pointed directly at this issue in their famous paper [22], 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 5.4 (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 $V_{\text{cross}}(t)$: the volume of the cross-section at time t,

[15] This is the BBS model, which will be discussed later.

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 5.8. 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.

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 [23] for a technical discussion of how this phenomenon has historically played out in computer architecture. Steele and Sussman [24] for another implementation. Future work will involve understanding this quantum boundary better to develop better high-level language CPU architectures. [16]

Remark 5.9 (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, would require a mathematical insight worthy of a Fields Medal. So with regards to the EVT Hypothesis, here I must stop, as this topic is a vast metamathematical maze and I wish to stick to the specific properties of twin helices for the remainder of the paper.

5.6 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, but as groupoids, they allow for a more flexible, local symmetry structure: instead of a single global group action, we have objects and invertible morphisms between them.

Lie groupoids provide a natural algebraic and geometric framework for describing the partial, local, and smooth symmetries that arise in fluidic and geometric computation. Where groups suffice for symbolic computation, Lie groupoids generalize these ideas to capture the location-dependent logic and dynamical transitions of fully geometric machines. 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.

When dealing with a groupoid, it is natural to ask what its fundamental invariants are. In the context of KTH, or indeed any helical solid, we arrive at a conclusion that many category theorists may find surprising: the winding numbers are *baked into* the geometry. These winding numbers are

[16] "Dude, why is the Python code running slow?" "Aha, well you see, the quantum universe is actually made of these computational glued together helices..."

not abstract group-theoretic constructs, but rather explicitly parametric homotopy classes "decorated" by the geometry itself. If one follows a path from the apex of one unit helix to the apex of the other, the total number of twists in each chirality defines the fundamental groupoid of the space—there is no freedom for deviation or continuous deformation beyond that imposed by the geometry.

Definition 5.4 (Decorated Fundamental Groupoid)**.** 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:

- **Objects:** Points $x \in M$ (as in the classical fundamental groupoid).
- **Morphisms:** 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.
- **Composition:** 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).

Remark 5.10. 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 [25], and Fong [26]). In particular, Baez's [27] 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.

In a GCM, the fundamental groupoid is not merely a topological invariant, but a combinatorial-topological hybrid: it records both geometric (number of twists) and logical (chirality) data.

Let M be a manifold, and let $\mathcal{H} = \{(\tau_i, \chi_i)\}_i$ denote the set of helices with parameters τ_i (number of twists) and χ_i (chirality) for each helix i. The decorated fundamental groupoid is defined as

$$\Pi_1(M, \mathcal{H}) = \{\text{homotopy classes of paths in } M \text{ decorated by } (\tau_i, \chi_i)\},$$

with composition given by concatenation of decorated paths, and relations arising from the gluing of types.

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.

For example, in the fully symmetric regime (V_4), the average winding number over all geobit types is $(0\times2+2\times2)/4=1$. 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 5.5 (Homotopy Computation)**.** Let $\mathcal{M}$ be a GCM, and let $\mathcal{S}_{GCM}$ be its state space. A **homotopy computation** is a decorated path $\gamma : [0,1] \to \mathcal{S}_{GCM}$, together with a sequence of computational (or geometric) operations, such that:

- The path encodes the evolution of states under the dynamical laws (flows, diffeomorphisms, group actions) of the GCM.
- Homotopies between computational paths correspond to equivalences or deformations of computations, preserving key invariants (e.g., outcome, topological class).

It is often observed that the fundamental groupoid of a contractible manifold is trivial in the classical topological sense—it wishes to live inside an ∞-category. But despite the fact that KTH is homeomorphic to a 3-ball, the introduction of global twist via topological deformation destroys triviality of the underlying homotopy classes one would find on the surface of a sphere. The fundamental groupoid corresponds now not a topological invariant, but a number. In this manner, the individual homotopies become not a path to a proof, but a path to a computation. In this space, morphisms between points become self contained FSS's that compose in total to become the overall calculation. Via the Howard-Curry Correspondence, which equates types, proofs, and programs, I justify this as valid categorical construction.

However, in the GCM lattice setting, the situation becomes even richer: the helices communicate via vector flows, but the totality of all helices is not, in general, a single connected geometric or topological space. Instead, the lattice should be viewed as a network of interacting groupoids—each encoding its own computational and geometric invariants, but coupled through the flow-based communication. This layered structure of a FSS is fundamental to the architecture of geometric computation.

Thus, even the most trivial topological spaces can host nontrivial groupoids when "decorated" by twists or combinatorics that form new habitat for mathematical structure, not requiring the machinery of ∞-categories, but being enriched by the underlying twisted geometry. The fundamental groupoid acts as a digital invariant, labeling computational morphisms within the groupoid, and serves as both a memory of past transformations and a structural fingerprint of the system's evolution. I understand this construction may be difficult for longtime practitioners of algebraic topology to swallow, but the author feels that as far triviality goes, the playground matters as much as the player.

Conjecture 5.5 (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 computations,
so that the groupoid serves simultaneously as a classifier of homotopy types and as a substrate for geometric computation.

Summary: *The world of decorated homotopy types—that is, spaces enriched with computable or combinatorial structure—is faithfully encoded in the world of decorated ∞-groupoids. The classical Homotopy Hypothesis is a limiting case: the general correspondence unites topology, logic, and computation in a single invariant.*

Remark 5.11. 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 are not mere static invariants, but may encode intrinsic computational or dynamical structure. A bridge between geometry, logic, and computation that Grothendieck's original vision only partially foreshadowed.

Now, with regards to the Lie part of groupoid. Let us explicitly denote the manifold encoding these variables as:

$$M_{\text{core}} = \{\chi\} \times \{\omega\} \times \{\phi(t)\} \times \{\phi'(t)\}$$

These are the discretely defined "core" variables. Each element of the Lie groupoid corresponds to a permissible local transformation or symmetry, defined explicitly by diffeomorphisms between open subsets of this manifold M_{core}. Formally, let the Lie groupoid of the core quasi-fluidic variables be defined as:

$$\mathcal{G}_{\text{core}} \rightrightarrows M_{\text{core}}$$

with source and target maps $s, t : \mathcal{G}_{\text{core}} \to M_{\text{core}}$, and a smooth composition law:

$$(g, h) \mapsto gh, \quad \text{whenever } s(g) = t(h).$$

This groupoid encodes transitions between logic and routing states via smooth, symmetry-based transformations and continuous or discrete rotations in ω and flips in χ, controlled by the flow variables ϕ, ϕ'.

The structure of $\mathcal{G}_{\text{core}}$ is finite-dimensional, smooth, and fully explicit. It is also algebraically dense, embedding naturally into a sufficiently high-dimensional Lie group due to the continuous nature of flow rate and orientation. Yet, this Lie groupoid is vastly simpler and more manageable than the full Lie groupoid structure of FHCM with all analog/physical variables which is most definitely infinite dimensional. Thus, the core FHCM Lie groupoid provides a rigorous yet manageable geometric algebraic framework for quasi-fluidic computation, neatly capturing the fundamental structure necessary for explicit mathematical and computational analysis.

5.7 Fiber Bundles and Moduli Spaces on Chiral Manifolds

Regarding triviality and computation in helical fiber bundles: it is true that the fiber bundle structure of a single twin helix manifold is trivial. The KTH is globally diffeomorphic to a cylinder $S^1 \times D^2$; more concretely, one can construct the fibration by attaching (gluing) along both sides of a disk, and then following the helical parameters.

This apparent simplicity, however, is misleading from a computational perspective. The triviality of the bundle means there is no topological twisting or obstruction (in contrast to, for example, the nontrivial Hopf fibration). Yet, when geometric computation is introduced—through operations on

chirality, orientation, and flow—the computation is encoded not in the topological class of the bundle, but in the dynamics and group actions imposed by gluing, deformation, and parameter manipulation.

It is precisely this triviality that enables a clean separation of "hardware" (the geometric manifold) from "software" (the computational process). The rich computational behavior of the system arises not from topological complexity, but from controlled geometric manipulations within an otherwise simple topological scaffold. In this sense, the triviality of the fiber bundle can be a saving grace.

As a practical analogy, recall that "jet spaces" in differential geometry record all derivatives of functions along a submanifold. For a trivial bundle, jet calculus can be applied directly: the absence of twists or singularities makes the derivative structure simple and computable. By contrast, in the Hopf fibration or other nontrivial bundles, holonomy and monodromy effects introduce quantum-like behavior and computational complexity—sometimes even leading to noncomputability.

For readers more comfortable with moduli space language, consider the two Archimedean spirals from Lemma **??**. The moduli space of the full 4D KTH is extraordinarily large (requiring, in principle, a Hilbert space due to the infinity of possible cross-sectional 1-manifold joins), so it is pragmatic to work with toy models and restricted cases.

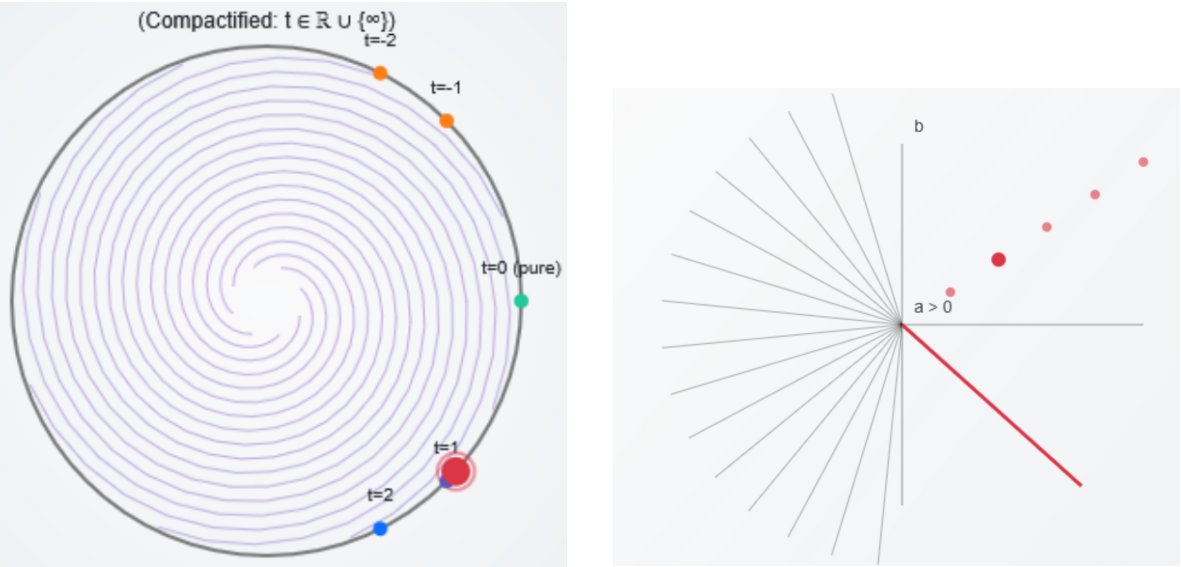


Figure 7: Visualization of the moduli space for two spirals with $|a| \div |b|$. On the right, the red line encodes an instance of an adjustable ray for right handed spirals. The red dot on the compact moduli space on the left corresponds to the phase shift. The equivalence rays opposite in grey encode possible left handed spirals. Note the singularity at $b = 0$. The two rays cannot meet.

Lemma 5.6. *Bifurcation of Moduli Spaces in Chiral 1-manifold Lemma. Two Archimedean Spirals sharing opposing chirality 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 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 **??**. But now it is done in reverse. It is not possible to construct a smooth moduli space that satisfies both of these spirals, as doing so requires the opposite of what we previously did, rather than trying to form a smooth 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 moduli space, one only need 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. Herein lies what I believe to be the harsh truth of what a cellular automata is *truly* made of, underneath that black and white screen we see.

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 5.7. ***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 KTH 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 entirely from the abelian group action, not from geometric singularities or non-abelian topological defects.

The central point underlying this is as follows: the topology of the moduli space associated with the first half of KTH_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 information in manifolds. Specifically, the ability for information to transition between states while remaining bifurcated by an intrinsic distinction could serve as the underlying principle for the emergence of geometric computation.

5.8 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 algebraic geometry and much of pure mathematics. Yet, as I have argued, maximal algebraic symmetry inevitably destroys geometric expressiveness—the very quality needed for a dynamical system in the quasi-fluidic regime, where symbolic encoding of state is essential. In fact, any form of computation requires at least some modicum of expressiveness, because without it, one is left only with propositional logic. Helices can dance with zeros or ones, but a sphere will never lie.

Remark 5.12. 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 5.13. 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, I 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—perhaps infinite-dimensional—an 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 6.6.)

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.

Though I have refrained from invoking specific biological or physical examples here, understanding where and why helices give way to other forms remains a promising direction for research in computational chemistry and biology. Author recommends readers see Chouaieb et al [29] for a background on the role of helices in nature and their emergence. Specific attention should be paid to figure 1 and 3—a telling sign of what this moduli space might look like. Likewise, investigation should be done into the niche body of research from Japanese geometers on Killing vector fields and their helical behavior. See Adachi [30] 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 [31]) on transcendental curves—waiting for someone to notice that helices and hyperboloids speak a universal geometric language.

It is here that we now must stop and examine the cave which we have tunneled into. It is a beautiful cavern with crystal clear flowing rivers and rich spiraling rock formations, but further exploration is dangerous and requires specialized caving equipment. We will move on to more practical considerations from here on out.

5.9 Spooky Action Up Close

What would a KL actually be able to practically compute reasonably better than a classic computer? Engineering problems in designing such structures at scale aside, consideration yields a rather remarkable conclusion.

Proposition 5.8. *A KL can perform quantum-like computations without entanglement or superposi-*

tion, 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 KL can achieve "quantum-like" expressiveness by encoding highly correlated, non-separable state information directly into their 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 KL, 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 KL 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 in character: 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.

In summary, GCMs achieve quantum-like parallelism and expressive power through deterministic, geometric mechanisms, and not through quantum phenomena per se. □

Regarding practical speedups, GCMs may offer computational advantages in the following broad classes of problems:

Symmetry-Constrained Problems. These 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 [32]) 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. This class 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 [33]) 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. Many CSPs 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 [34]). 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 5.14. **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 its notorious difficulty. The solution space of NS (and by extension—P vs. NP) 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 micro-expressiveness.

As a consequence, the act of solving it is 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 5.2. 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 [35] 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 5.6 (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 $O(|S|)$. Concrete example is below.

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]
```

Table 7: 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)

```
    }

    // 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 KL 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 5.9 (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 phenomena, 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 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 possibly more of an ultra-powerful edge case at the end of a continuum. See Tang [36] for a case study that demonstrates why clever deterministic approaches can perform on par with quantum computers.

Remark 5.15 (Programming Language Outlook). Developing a programming language suited to geometric computation remains a substantial challenge. However, physical realizations of non-Von Neumann computing are not without precedent. While these machines build by MIT hackers of yore may be mostly forgotten, the "tagged token" dataflow computer architectures of the 1980s [37] is an instructive example of deterministic, non-standard models. Comparable experimental attitudes informed projects like Butterfly LISP, an esoteric dialect engineered for parallelism across heterogeneous architectures such as the BBN Butterfly and Symbolics LISP Machine [38]—demonstrating explicit parallelism in software, even if the hardware was only partially faithful to the ideal.

For an actual language, I 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 [39]. Future progress in this direction will likely require close collaboration with compiler experts and programming language theorists.

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 still 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 a way that is intuitive and bridges group theory, topology, dynamical systems, and category theory. [17]

5.10 Chaos Unleashed

After assembling KTH in the abelian, subrecursive regime—where geometric computation is governed by composable, finite operations on computable primitives—we enter the fluidic phase space. Here, the helix structure is treated as a fiber bundle $\pi : E \to B$, where the base B is a computably generated manifold (e.g., subrecursive or piecewise-constructible), and the fibers carry smooth internal state variables. Computation can be modeled as a flow along a connection ∇ on E, defined in a synthetic differential geometric setting, allowing for infinitesimal evolution without reference to $\mathbb{R}$. This regime breaks the join symmetry; join operations become path-dependent, curvature encodes memory, and noncommutativity emerges via holonomy and Lie groupoid structure. Computation thus proceeds by the progressive internalization of curvature, transforming symbolic joins into geometric flows.

In the language of Lemma 3.5 this is when $\mathcal{G}(J)$ almost completely subsumes $\mathcal{A}(J)$. Via the chart of n-logics we flip from a quasi-fluidic regime dominated by maximal abelianism to a purely fluidic computational regime dominated by non-abelian structure. Though non-abelian groups are present from the symbolic computing mode they are simply extensions of the initial abelian group. 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 Section 3.6, there is a degenerate case corresponding to a KL 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 KL—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 5.10 (Computation–Communication Tradeoff in GCT)**.** *There exists a fundamental tradeoff between internal computability and external communicability (accessibility) in geometric computation.*

To formalize: let $\mathcal{G}(J)$ denote a GCM with quantization parameter J, and let $\mathcal{N}(\mathcal{G})$ be the number of connections required to simulate a Turing machine. As $J \to \infty$, the information-theoretic capacity of $\mathcal{G}(J)$ diverges, but $\mathcal{N} \to 0$.

Remark 5.16. This phenomenon is directly analogous to the measurement problem in quantum theory, and to limits on information extraction from chaotic classical systems. In practice, every real physical system is subject to limits (such as the Planck length or thermal noise), so infinite computation is always an idealization. A fully continuous GCM is, in principle, an oracle of infinite information, but loses all capacity for communication, control, or interaction. Only by re-imposing quantization do we recover a system that can read, write, and exchange information.

A classic lesson from computer architecture is Amdhals law. Scaling up computational capacity whether by increasing memory, processing units, or geometric lattice size runs into an ultimate physical

[17] 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.

barrier: the speed of signal propagation. Even if each individual GCM (as a idealized "mini-FPGA") can be made arbitrarily powerful via increasing the state group of each in a KL to Lie size, the maximum rate at which the entire system can synchronize is limited by the time it takes for information to traverse the system's diameter. In an idealized setting where communication occurs at the speed of light c, the maximum global clock frequency is bounded by

$$f_{\max} \leq \frac{c}{2L}$$

where L is the physical diameter of the computational lattice or network. As L increases, $f_{\max}$ falls inversely.

In an idealized geometric computation system, the ultimate limit to system performance is set by the speed of communication. No signal can travel faster than the speed of light. Thus, the maximum clock frequency for global synchronization scales inversely with the system's physical diameter. This mirrors the constraints seen in classical computing hardware, where parallel scaling is ultimately limited not by local processing power, but by the fundamental laws of physics. A KL does not offer infinite power combined with infinite expressiveness.

Table 8 quantifies a hierarchy that demonstrates a natural algebraic-geometric progression inherent in GCT. From discrete symbolic logic encoded in the simple abelian groups and low-dimensional spaces, through progressively richer non-abelian algebraic structures, one eventually arrives naturally at structures analogous to gauge theories and topological quantum field theory. **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 8: The Full View of GCT and Topological Computation.

Group Type	**GCT Model**	**Topological Field**	**Example Groups**	**Embedding Space**
Abelian	Symbolic Mode KL	Finite group TQFT	$V_4, (C_2)^3$	$SO(3), SO(4)$
Maximally abelian	Quasi-Fluidic Mode KL	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	Full-bore TQFT (Yang–Mills, Chern–Simons)	Manifold bundles	$SU(2), SO(N)$

5.11 A New Frontier and an Algebraic Gap

At present, no known algebraic structures fully capture the behavior of computation in GCMs across both symbolic (discrete) and fluidic (continuous) modes. The theory of Lie groupoids and their relatives currently marks the "event horizon" of mathematical understanding in this direction. Beyond this, no general framework of morphisms or categories is available that accurately bridges a GCM's geometry and its computability.

Indeed, as one transitions from discrete to quasi-fluidic regimes, finite algebraic structures quickly break down; the underlying symmetries become too flexible, and the combinatorial invariants dissolve into a sea of continuous dynamics. This suggests that an approach rooted in dynamical systems theory—rather than classical algebra—is required in such regimes. Consequently, the use of the term "algebra" here is necessarily loose and aspirational.

Below, I summarize representatives from several major algebraic families and outline their respective "failure modes" in the context of GCMs:

Boolean Algebra underlies classical digital logic, but it fails to capture the continuous deformations and topological richness inherent in GCMs.

Heyting Algebra generalizes Boolean logic for constructive reasoning and partial orders. Though the flows within a GCM may resemble a partial order, Heyting algebra does not directly encode computability or the geometry of information propagation.
Lie Algebras are powerful tools for analyzing continuous symmetries and local properties of manifolds via tangent spaces. However, GCMs are fundamentally global, topological objects, and Lie algebras assume local linearization.
Quantum-style Algebras (Von Neuman, C*) These non-commutative algebras excel at describing indeterminacy, entanglement, and operator structure. But GCMs, as formulated here, are fully deterministic; there is no need for probabilistic or Hilbert space-based frameworks.
Algebraic Topology is a crucial piece of the puzzle no doubt when classifying and understanding the global topological properties of GCMs. Nonetheless, by itself, it lacks a calculus for the computational aspects intrinsic to these objects.
Group Theory and groupoid structures are central to the construction of GCMs, capturing symmetries and allowed transformations. Yet, pure group theory has no sense of the computational properties necessary for Turing completeness; it needs to be layered on top.

Remark 5.17. What might a language for this kind of computation actually look like? Purely symbolic, algebraic, topological, or analytic frameworks—taken in isolation—are inadequate. Any successful theory must simultaneously address symbolic and information-theoretic composition of states, geometric flow and deformation, topological boundaries and manifold joins, and the impact of computational update laws on geometric structure.

Attempts to model the computational structure of GCMs using traditional algebraic systems encounter a fundamental obstacle: the nonlocal nature of constraint propagation across helices. This structure resists both local linearization and naive discretization.

What is needed? A successful mathematical branch for geometric computation should support at least four critical capabilities:

1. **Continuous transformation:** The language must represent continuous geometric changes (e.g., helix tube deformations, chirality flips, twist rate variations).

2. **Local-to-global behavior:** It must faithfully track how local computations propagate through space and induce global effects.

3. **Structural invariance and transformation laws:** Orientation, chirality, and topological identity must be preserved during flow—or, when altered, their transformation laws must be fully expressible.

4. **Rich morphism structure:** The formalism should provide invertible morphisms corresponding to computational transformations, which need not be group actions but should be composable within a suitably rich structure (e.g., a higher category or groupoid).

The following areas of mathematics come to mind as to what might *actually* work:

Sheaf Theory and Stack Structures — help formalize the idea of local-to-global consistency which is the core problem in GCT. One could cover a 3D/4D manifold with open sets $\{U_i\}$, each with a unit helix state keeping track of its chirality, volume, number of twists, etc. A presheaf assigns to each open set a set of computational states, and to each inclusion $U_i \subset U_j$, a restriction map. Sheaves could ensure consistency and help capture distributed computation via continuous updates since different regions perform local ops. In the fully fluidic regime, a GCM's "chaotic geometry" is modeled as a smooth moduli stack parameterizing all possible internal geometric configurations, with computation realized as a smooth path or flow through this stack, governed by differential equations rather than discrete update rules.
Topological Field Theories (TFTs) — associate algebraic data to topological spaces (see Lurie [40] for an overview). It could model the evolution of a GCM as a path in the configuration space of the manifold, and assign computational observables to each step. This may help recover a "topological conservation of information" principle for fluidic computation as topology remains stable under continuous deformation giving a way to classify computable operations in a geometry-preserving manner.

Synthetic Differential Geometry and Jet Bundles — allow tracking of not just positions but all derivatives up to order n. This is ideal for modeling how twists, flows, and orientations deform over time and space. It encodes continuous flows and geometric deformation in one object. GCM states can be modeled as evolutions over sections of fiber bundles. Computation then becomes differential constraints on how these sections evolve similar to solving fluid-flow PDEs—but instead with a new kind of "synthetic differential computational jet calculus" over the manifold.
Fluidic Type Theory (FTT) — This theory [18] posits a direct correspondence between types and decorated geometric manifolds, such that:

Types — are classes of decorated geometric manifolds (e.g., helical, stratified with computational data).
Terms — are specific geometric configurations of a manifold with particular parametric and computational attributes.
Id types — correspond to homotopy computational paths: explicit, decorated paths (including geometric and computational operations) connecting configurations.
Π-types — model decorated fiber bundles over moduli spaces of morphisms
Σ-types — are composite geometric assemblies or glued manifolds with computational boundary data.

FTT, in this sense, unifies aspects of Homotopy Type Theory, Differential Geometry, and Computational Topology—anchored not in abstract sets, but in real, constructible manifolds with computational semantics. The "extended homotopy hypothesis" conjectured herein suggests that such a theory could, in principle, classify and verify the behavior of GCMs both numerically and structurally, opening the door to a new computational mathematics.

Remark 5.18. The type theory proposed here should be confused with *Geometric Type Theory*. Proposals for such a theory [19] are related to geometry of the coordinate-free variety, of which we cannot practice. Naming has been chosen to avoid collisions. For those interested, see recent work by Von Branitz and Buchhholtz [41].

There are of course many other avenues to explore. While as of now it seems not completely necessary for basic GCMs, some application of higher category theory [20] will be necessary considering the interpolation between many different areas of math to describe geometric computation. A Donaldson-style moduli space and cohomological approach [42] also comes to mind as promising, but for the time this list must suffice.[21]

Remark 5.19. 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 I 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—despite the fact that geometric computability theory owes so much to his vision of mathematics as a seascape of ever-expanding generality. [22]

[18] Admittedly, for the sake of intellectual honesty I must confess that this field of mathematics doesn't technically exist yet, but I am acting like it does nonetheless for convenience. Likewise we might also use the term "Bordism Type Theory", if a general purpose implementation related to Miranda at al.'s work is desired. It is open at this point.
[19] Nlab states—"At present it is not clear exactly how such a type theory should be defined."
[20] I am indebted to Emily Riehl for her excellent video series/lectures on the subject.
[21] Please note that I withhold examination of the work of Prof. Donaldson here not out of lack of importance, but because the formation of the a "Computational Donaldson Theory" requires an *entire paper onto itself* to do his great work justice, in addition to a computational analogue of Floer.
[22] Though I will admit I now understand his distaste for physics, even if I personally do not share it.

6 Categorical, Philosophical and Practical Considerations for Geometric Computability

We now turn to the higher-level philosophy of GCMs and their implications, beginning with computational isomorphisms known from classical theoretical computer science.

Conjecture 6.1 (The Geometric Computability Thesis)**.** *For every GCM $\mathcal{G} = (M, \Sigma, \Phi)$ with sufficient geometric and topological richness, there exists a Turing machine T such that the symbolic outputs of T are isomorphic to the boundary outputs of $\mathcal{G}$ under Φ, and vice versa.*

Stated informally: *Every computational process that can be computed by a TM can be computed by a KL.*

This is, in spirit, an analogue of the Church–Turing Thesis, transplanted into the realm of geometric computability. 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 6.2 (Geometric Logic Isomorphism)**.** *For every Turing machine T, there exists a KL such that*

$$\mathrm{eval}_{TM}(x) \cong \mathrm{eval}_{KL}(x)$$

where the correspondence is structure-preserving (i.e., it is an isomorphism of computational processes, not merely of outputs).

This second conjecture aspires to a deeper, type-theoretic structural isomorphism in the spirit of the Curry–Howard correspondence: strict equivalence between symbolic computation and geometric computation, with the mapping respecting computational structure at every level. As with much of Curry–Howard–Lambek theory, the precise formalization of this correspondence in the geometric setting remains an open challenge. The notation here is necessarily somewhat informal.

The study of homotopy computation will once day help illuminate this issue, and again I stress that all topological spaces have these homotopies via the fundamental groupoid, but most lack the correct symmetry for actual computation, compared to a transcendental curve such the helix. Algebraically however, we are in a territory that is only beginning to be explored. The study of these "homotopic computational paths" may one day clarify the landscape, but the full machinery of a "fluidic/bordism type theory" (capable of understanding the symbolic composition of GCMs) has yet to be developed.

6.1 Constructing Categories

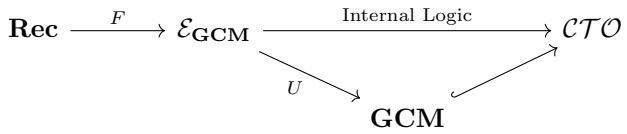
While the author is not a specialist in category theory, it is clear that the algebraic enrichment required for this framework demands categories capable of encoding both geometric and computational structure. To this end, we introduce the category $\mathcal{ESM}^+$ ("Embedded Smooth Manifolds with Enrichments") as the natural habitat for all GCMs. For background, see Blass [43].

Definition (Ambient Category). The master category $\mathcal{ESM}^+$ is defined as the symmetric monoidal category whose objects are geometric, smooth, or computational manifolds (including all GCMs), and whose morphisms are smooth maps or structure-preserving transformations that respect both geometric and computational enrichment. The symmetric monoidal structure encodes the physical and logical composition of systems (e.g., juxtaposition or parallel flow).

Within $\mathcal{ESM}^+$, we distinguish the following subcategory:

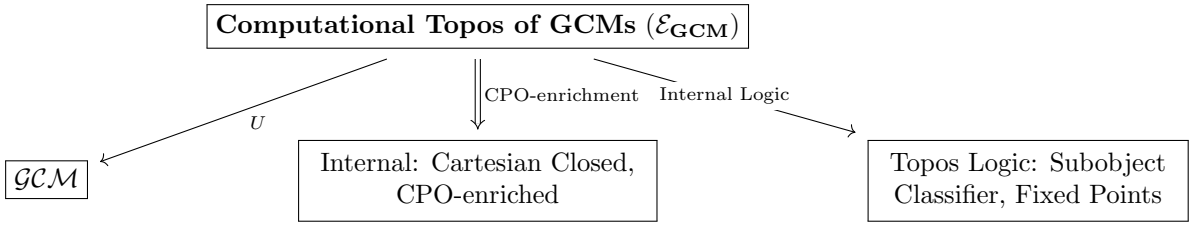
Definition 6.1. A *computational topos* ($\mathcal{CTO}$) is a cartesian closed symmetric monoidal category $\mathcal{E}$ whose objects are GCMs or related computational geometric structures, and whose morphisms preserve both geometric and computational structure (including, for example, groupoid decorations or logic assignments). The internal logic of $\mathcal{E}$ models computability—whether partial or total, binary, fuzzy, or otherwise.

The monoidal product (typically denoted $\otimes$) captures the parallel composition or aggregation of independent GCMs, while the cartesian closed structure enables internal logic, function spaces, and higher-level constructions. These categorical properties are essential for modeling both the logical and physical composition of geometric computation.



Rec - Category of all recursive computations
F - Functor embedding Rec into $\mathcal{CTO}$
$\mathcal{E}_{GCM}$ - The computational topos of GCMs
U - Underlying geometric manifold functor
Internal Logic - The internal language/topos logic

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. Establishing and characterizing such enrichment in the context of geometric computation is left to specialists in the field. See below for a probable outline.



Geometrically Computable Manifolds — (**GCM**) and the Computational Topos ($\mathcal{E}_{GCM}$) of GCM. A computational topos $\mathcal{E}_{GCM}$ is a cartesian closed subcategory equipped with an internal logic of computation, supporting fixed-point and recursion theorems. GCMs, be they chaotic process or discretely chained in a KL, will have objects $\mathbf{E}_{GCM}$ are specified by boundary data (input/output), and morphisms are internal symmetries or transformation flows, closed under composition. Composition will involve the chaining these geometric transformations or the composition of various GCMs in sequence into a KL.
Sh(x) — The category of sheaves will help us understand local to global behavior of the geometry. The category is a space that is rich in limits, colimits, and functorial constructions. Together it gives a way to treat geometry, logic, and computation within a unified categorical glue language.
$\mathbf{Man}_\infty$ — The category of all smooth manifolds, which serves as the geometric "building block" for GCMs. While not every smooth manifold need be a GCM, this category includes all possible n-manifolds, which may serve as cross-sections, spatial domains, or higher-dimensional GCMs. Morphisms are smooth maps such as diffeomorphisms, immersions, embeddings, and related structure-preserving maps.
Fix(f) — Both chirality and orientation are observer-dependent phenomena. To account for this, we introduce fixed reference frames: a distinguished hyperplane (or set) of fixed points, representing "observation structure" within the category. Morphisms in this context are required to preserve these fixed points. Since chirality is observer-relative, we may formalize this via a contravariant functor:

$$\mathcal{V} : \mathcal{O}^{op} \to \mathbf{GCM}$$

where $\mathcal{O}$ is a category of observers or reference frames, and $\mathcal{V}$ assigns to each observer a GCM as seen from that frame.
Grd — The category of groupoids (with **Grp** as a subcategory) encapsulates the symmetries and allowable transitions—logical, geometric, or dynamical—of GCMs. Groups model global reversible symmetries (e.g., V_4, D_4), while groupoids allow for partial or local symmetries, where not all elements are connected. Objects in **Grd** are finite groups or Lie groupoids acting on spaces or sets; morphisms are group or groupoid homomorphisms. Every step of geometric computation in this framework is fundamentally a group or groupoid action on the current state.
$\mathbf{S}_{\mathbf{GCM}}$ — The category of state spaces $\mathbf{S}_{GCM}$ will handle the logical the configuration space contained in $\mathcal{CTO}$. The existence of group actions and computation must also be accounted for. Encompasses

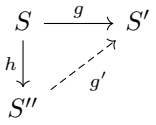
all possible logical/geometric states of a GCM. Each GCM has a finite (symbolic) or infinite (fluidic) state space, determined by its geometry and group actions. It encodes "where" in the computation one is. Objects are sets or manifolds parameterizing possible GCM configurations. Morphisms are maps (possibly group-equivariant) between state spaces, corresponding to allowed transitions. For symbolic regime, a finite set with group action and for fluidic, a manifold with a diffeomorphism group. We define a functor

$$\mathcal{C} : \mathcal{GCM} \to \textbf{BitStreams}$$

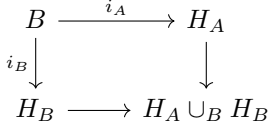
which interprets each object's discrete symmetries as initial conditions, and morphisms (flows) as state evolution.

$\mathbf{Vect_k}$ — The category of vector spaces provides the linear structure necessary for modeling flows, quantum analogies, and information propagation. In symbolic computation, encodes possible linear combinations. In fluidic computation, the tangent and cotangent spaces at each point in a manifold are vector spaces critical for defining flows. Objects are vector spaces over a field k (often $k = \mathbb{R}$ or $\mathbb{C}$). Morphisms are simply linear maps between vector spaces.

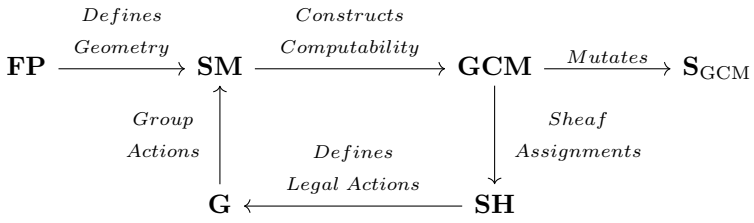
Commutative diagrams can be used to effectively show the action of a GCM where: S, S', S'' are state spaces, and g, h, g' are groupoid morphisms or actions.



As well as a pushout map for the topological construction of KTH itself.



The following is a "meta-map" (not a commutative diagram *per se*) that overlooks the landscape of GCT via its categories and their functors. Designed to show the input of fixed geometry → "central cycle" of computation → to stateful change.



Conjecture 6.3. *Geometric Fixed Point Hierarchy Conjecture. There exist faithful functors between the categories of observer frames, geometric fixed points, and computational fixed points, such that invariants at one level are preserved at the next. The precise nature of these functors—whether full, faithful, adjoint, or otherwise—remains an open question for future categorical analysis. Or, more formally:*

$$\mathcal{O} \xrightarrow{faithful} \mathcal{G} \xrightarrow{faithful} \mathcal{C}$$

such that the structure of fixed points at the observer level is preserved and reflected in the geometric fixed points, which in turn induce fixed points for computational processes defined on or relative to the underlying geometry. Fullness may not always hold globally, but is expected in restricted or "well-behaved" subcategories.

The study of fixed points has been central to mathematics for nearly a century, particularly at the intersection of topology, geometry, and algebra. Early work by Solomon Lefschetz in the 1920's provided algebraic tools for detecting fixed points of continuous maps via what is now known as the Lefschetz fixed point theorem. In the 40s, P.A. Smith developed a foundational theory relating group actions on manifolds to the structure of their fixed-point sets (now called "Smith theory")

which remains fundamental in transformation group theory. By the mid-20th century, the framework of equivariant cohomology, pioneered by Borel, Atiyah, and Segal, offered a systematic language for encoding symmetries and their influence on topological invariants, especially fixed points and invariant subspaces. These developments are all related by a deep connection, that the existence and structure of fixed points under symmetries often reflects, and sometimes dictates, the global topology of the underlying space.

The current work extends these classical perspectives to the setting of geometric computation, positing a natural hierarchy between three kinds of fixed points:

1. Observer-invariant reference frames
2. Geometric/dynamical fixed points arising from diffeomorphisms or group actions
3. Computational fixed points, as in recursion theory or logic.

I conjecture that these layers are connected by functorial correspondences, reflecting a geometric-computational analogue of the classical theory. In particular, invariant hyperplanes or subobjects in $\mathcal{O}$ induce fixed sets in $\mathcal{G}$ (Smith and Lefschetz), and these in turn anchor or determine the existence and interpretation of computational fixed points (Kleene or domain theory) in $\mathcal{C}$. So in the end, the structure of geometric computation appears to be, as the saying goes, "fixed points all the way down".

6.2 Implications for Reversible Computing

GCT opens an avenue for *reversible computing.*[44] In reversible computing, computation works more like the laws of physics, running both backwards and forwards. This is obviously 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 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 KTH 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.

6.3 Comparison to Blum-Shub-Smale Model

A foundational precedent for my approach can be found in the Blum–Shub–Smale (BSS) model [45] 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 ("BBS 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 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 model 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 in a non-Von Neumann style.

Where the BSS machine assumes access to uncountable numbers of real values, a GCM lattice assumes access to geometric substructures and continuous group actions as primitive. Computation arises not from the manipulation of symbols or numbers, but from the controlled evolution of geometric state, driven by fluidic flows, chirality shifts, and manifold deformations and so on.

This moves us closer to a physical theory of computation, where space, time, and shape are no longer external to computation—they simply are *the* computation.

6.4 Computing the Uncomputable?

As a short aside, it is natural to ask whether geometric or quasi-fluidic computational models might transcend the limits of classical computation, such as those demarcated by the Busy Beaver function [46]. 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 geometric analog/chaotic 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.

6.5 Relationship to Topological Kleene Field Theories

Miranda, Peralta, and Gonzales [47] have recently introduced Topological Kleene Field Theory (TKFT), which advances the 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, 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 6.2 (Geometric Fluidic Computational Bordism)**.** Let M_0 and M_1 be n-dimensional geometric computational manifolds (GCMs). A *geometric fluidic computational bordism* is a smooth $(n+1)$-dimensional manifold W (possibly with additional structure: flows, group actions, computational data) such that $\partial W = M_0 \sqcup M_1$. The bordism W represents the explicit geometric realization of a computational transformation, encoding the process or "flow" from M_0 to M_1.

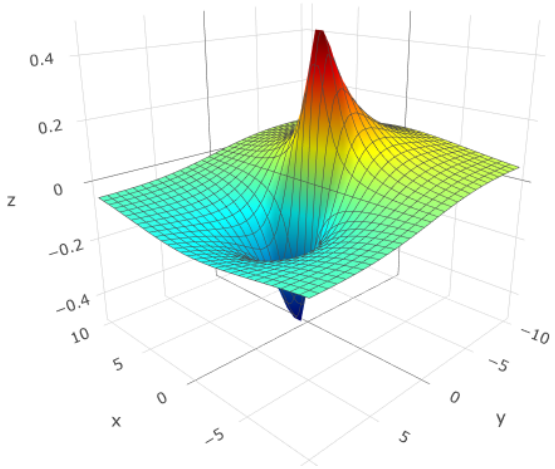
In GCT, a more intrinsic and lightweight perspective is natural. Here, bordism is *internal* to the GCM: computation is realized as deformation or flow between different boundary states within the manifold itself. The "input" and "output" boundaries correspond to loci where topological flows implement 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 ($\mathbb{N}$), exploits the full power of recursive and automata-theoretic computation, and can recursively evolve its own computational history; nevertheless, it does not reach the uncountable or geometric-analog domain.

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

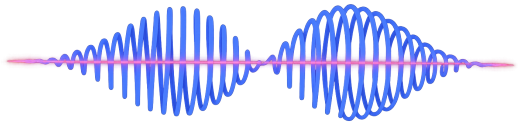
tasks. Thus, GCT is not a competitor to TKFT, but a complementary framework, with distinctive strengths and tradeoffs. In principle, both approaches can be used in concert to address fundamental problems in topological physics from different vantage points.

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.

6.6 Computational Gauge Theory and the Yang-Mills Mass Gap Existence



(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 8: Some Interesting Examples of Geometric Configurations Corresponding to Type 3 and Type 1 KTH.

The area between mathematics and physics has been a great source for fertile cross pollination of ideas with work from mathematicians like Kontsevich [48], Connes [49], and physicists like Witten [50] tackling questions related to quantization, fundamental force symmetries, and spin to name but a few.

Twin helix systems show up in a surprising number of physics systems. As an example—for any solution to the free-particle Schrödinger equation, KTH 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 "moving helical orb" in 8 resembling the Type 1 KTH. 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, Majorana fermions have been proposed to manifest as topologically protected twin-helix states in condensed matter systems [51]. 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. [52] 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 KTH_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 2-geobit state space, providing both a more granular symmetry classification and a richer foundation for both physical insight and computational applications.

Remark 6.1 (Dual Helicity Join Structures). 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 KTH 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**	**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 demonstrates 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. Quantum spin networks arise via coarse graining of the underlying micro-geometry of helices.

Remark 6.2. Although this paper does not 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 Sir Roger Penrose, John Baez, Carlo Rovelli, and other proponents of Loop Quantum Gravity. In addition, it closely aligns with the non-Markovian view of quantum mechanics espoused by physics philosopher Jacob Barandes [53]. [23] In this view, 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. Indeed—it is true poetry of nature that the geometric "DNA" of quantum mechanics might echo that of life itself.

Remark 6.3. Although 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 [54], 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 or even fully continuous (e.g., SO(2)). Thus, twistors become a distinguished phase within a far broader universe of transcendental, computational geometric structures—a special case of the general principle that geometry and computation are deeply intertwined. [24]

[23] We reject superdeterminism not only for physical reasons, but on cultural and epistemological grounds: any theory that precludes all genuine choice is incompatible with both the Rabbinical tradition and the mathematical spirit of independence proofs. Following the Sages, we interpret free will as path-dependent randomness—a process neither strictly determined nor wholly unconditioned, but always shaped by prior steps.

[24] 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 BBS model are, in reality, the prime examples of continuous algebraic computational flow.

As time is constrained, the example of a potential application of GCT I would personally would like to highlight would be that of the infamous Yang-Mills Mass Gap Problem. 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 [55].
Sphalerons: Unstable saddle points in electroweak theory [56].
Twisted solutions: Solutions over compactified or cylindrical spacetimes [57].
Witten's helical solutions: Helical structures in certain lower-dimensional reductions [58].
Non-abelian plane wave solutions: Chiral, twist-like structures in non-abelian gauge theory [59].

While this discussion does not claim to exhaust the intricate connections between geometric computation and modern field theory, it is worth highlighting the specific case of the Yang–Mills Mass Gap Problem. But, as illustrated in Figure 8a, 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.

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 suggests that geometric computability provides 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 [60]; for a modern perspective, see Donaldson [61].

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 level—full gauge regime allows for history-dependent, topological computation, where even the process of transitioning between states (as in sphalerons) becomes part of the logic.

Example 6.1 (The Computational Consequences of Non-Abelian Plane Waves). As an example to one solutions of the cited above, consider the appearance of non-abelian plane wave solutions as described by legendary Harvard physicist Sidney Coleman.

$$L = L_{\text{extr.}} = -\frac{1}{4}\text{tr}(F_{\mu\nu}F^{\mu\nu}) = \frac{1}{2}\sum_{a=1}^{n}\left[(E^a)^2 - (B^a)^2\right] =: \frac{1}{2}(E^2 - B^2).$$

This is discussed by Kastrup in [62] 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. 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. Thus, 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 may be lazy, but she is also expressive.

Table 9: Geometric Computation Regimes and Particle Physics Phenomena

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

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 KTH and focusing on the subgroups corresponding to Type 1, 3, and 4 KTH. **This choice was motivated, in part, by the formulation of the Yang-Mills mass gap problem itself.** 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 [63].

Remark 6.4. 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 6.5 (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 KL could serve as a fundamentally computable model of gauge field dynamics, where geobits play the role of discrete, intrinsic gauge degrees of freedom. [25]

Remark 6.6 (Issues with Hyper-Turing Complete computation over $\mathbb{R}$). It is at this point however a problem I have so far put aside rears its head—actually modeling the computation. Recall table 8. 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, categorical, or quantum invariants to encode information, and the

[25] "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 [64]

notion of computation becomes vastly more abstract. Whether any kind of programmable or scalable computational formalism can survive in this context remains an open question.

In principle, to implement the gauge regime, it could be modeled with a very powerful fluidic GCM that computes over $\mathbb{R}$, but as we know from the lessons of the BBS machine—this is a difficult beast to control. This is why again I mentioned synthetic, not regular differential geometry as the best course of action in the two uppermost regimes. Although the halting problem can technically be solved with an unrestricted approach over the reals, this involves the execution of borderline reality-warping computability hacks such as branch functions encoding oracle operations.

And even then, computations that are trivial in $\mathbb{N}$ can become impossible to decide in $\mathbb{R}$. Certain "gaps" or spectra might only be accessible or defined via limit processes or transcendental solutions that cannot be encoded finitely. If the only way to "compute" or even "define" the mass gap uses some non-recursive real, the problem is, in a sense, nonconstructive and incomputable. The BSS model over reals demonstrates that "hypercomputation" with real numbers without restrictions is vastly more powerful, and in fact can be quasi-pathological or even unclassifiable in the regular Turing-complete sense. This leads to a sad possibility:

Conjecture 6.4 (Lost Melody Conjecture 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:

1. *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).*

2. *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.*

3. *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.*

4. *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 that even if the entire mathematical apparatus was 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 6.7. Full treatment of the Lost Melody Theorem is beyond scope here, see Carl [65] for recent proof on the BBS model and Hamkins and Lewis [66] 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. [67] 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, a solution may still possible, laying right beneath the real numbers. In Yang-Mills theory there are four constants.

Gauge Coupling Constants - for $SU(3)$, $SU(2)$, and $U(1)$ and
QCD Vacuum Angle - the θ_{QCD} is constrained experimentally to be near zero.

These are typically treated as real numbers, but all these constants were determined empirically. They could be modeled as rational numbers with arbitrary precision. Upon researching, I do not claim this to be a completely novel idea. As Matthew P. Szudzik writes in, *The Computable Universe* [68]:

> *"The central problem is that physical models use real numbers to represent the values of observable quantities, but that recursive functions are functions of non-negative integers, not functions of real numbers. To show that a model is computable, the model must somehow be expressed using recursive functions. Careful consideration of this problem, however, reveals that the real numbers are not actually necessary in physical models."*

The computational properties of geometric models are determined not only by their topological or smooth structure, but critically by the choice of underlying number system. I might add that the closer one gets to the Planck length and then below, the worse $\mathbb{R}$ behaves. If the real numbers are too difficult to handle, but natural numbers lack richness needed to encode computation past gauge theories that correspond to abelian modes in GCMs, then computing over $\mathbb{Q}$ or its extensions (implicitly or explicitly) might just be the winning compromise.

If the issue of computation via construction can be sorted, as far as proving the Yang-Mills Mass Gap, I can only speculate on a high level road map.

1. Create a category/algebra to accurately map the emergent computational properties of manifolds that arise from GCT.
2. Formalize the GCM model as a computable fiber bundle via synthetic differential geometry.
3. Show it approximates continuum Yang–Mills dynamics in the scaling limit using a kind of "differential computational jet calculus".
4. Find a spectral gap in the model, as in, a provable "minimum energy" of certain transitions.
5. Then prove that this discrete mass/information gap survives the continuum limit.

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 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 3.6. This is the final entry at the top. Its geometric expressiveness—"Extreme", its algebraic closure—"Zero", less than trivial. 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 power, but at a terrible 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 6.8. While speculative connections between KTH_4's structure and fundamental particle physics or gauge interactions may seem tempting, I do not believe this to be the case. The most adventurous but conservative approach I am willing to 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. This connection positions geometric computational manifolds as elegant geometric analogues or realizations of discrete symmetry structures utilized in quantum field theories.

Indeed, there is something unique about the Klein four-group. It is the minimal nontrivial abelian group with full symmetry but no direction—every non-identity element is its own inverse. Our universe may be built from the smallest of groups, not just the richest ones. Einstein, Podolsky, and Rosen [69] may not have been wrong with their paradox, but rather proposed the right solution to the wrong question.

Disclaimer: The results presented herein **should under 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.

7 In Closing

Epilouge.
The year is 2153.
The central class hall of the physics grad students at Carnegie Mellon University. A man enters and looks at his classmates laptop.
"Man, you still using the old Helicycle 16C? Why not upgrade to the Helimax 512?"
"I don't know dude, I just don't trust the new non-abelian hardware for my daily driver. And I get that fuzzy chips are more efficient but it makes me feel weird when we have to turn on precision correction for certain things. I hear someone messed up the n-ary logic conversion in their code on their latest CFD assignment and got a D"
"Serves him right for using ChatGPT HV203-O on his homework. And yeah the new NVIDIA H700X series drivers for HGPUs aren't quite optimized yet but..."
A woman next to them chimes in, "Hey, word on the street is the dept head just got a grant to upgrade the server cluster to a new LIE-Compute 133X. I heard it can run 3-valued chirality logic natively."
"Nice! Hopefully they can afford the Groth-2560 sheaf accelerator coprocessor. My thesis on topological fixed-point computation is going to rock."
The professor enters: "Greetings class—hope everyone is excited, today we'll be going over proof of the Chern-Simons quantum hall effect..."

Indeed. One can dream. But, returning to all seriousness, at this point, the author must confess that as this paper draws to a close he is both equal parts unsettled and fascinated at the direction it has taken. While I take full responsibility for the fact that this paper, if proven correct, is indeed likely to cause at minimum, an epistemic shift of some non-trivial size in the computer science and quantum physics community, the most important purpose in writing this is to remind people that mathematics can never be a completely true reflection of reality. We are taught this in undergraduate calculus with Gabriel's Horn, and we shouldn't forget this lesson—even when formulating questions to the universe's darkest secrets.

This new subfield formed at the intersection of geometry, topology, and computability theory I have tentatively named "Geometric Computability Theory" in an attempt to capture the semantic richness while hopefully steering clear of any confusion with "Computational Geometry". I had originally thought one might find a sort of "computable geometric zoo", of which of many different classes of manifold were waiting to be discovered. But this may not be the case. Twin helices may be the first and last of their kind. Geometric computation requires a delicate balance, too much symmetry invites triviality. The idea that an entire subarea of mathematics purposely went out of its way to disguise itself in the form of an unusual geometric shape is equal parts astounding and somewhat disturbing. Whether or not there exists any other GCMs in the unmapped $\mathbb{R}^3$ jungle or dimensions *plus ultra* is an open question.

Problems like the Riemann Hypothesis may receive the lions share of attention when it comes to notoriety. Of the remaining 6 Millennium Problems Yang-Mills decidedly ranks in the lower half in terms of perceived academic interest, and among the 3 that deal with questions at the intersection of computer science and physics it might easily be considered last. Even if Dr. Tao's original focus was on a different question (Naiver-Stokes), Yang-Mills may be where the true demons lie in wait. Perhaps we should not be so surprised that it has joined the fray of the current academic foray around fluidic computation. After all, when it comes to fluid and electromagnetism, both are theories of vectors. Solving the mass gap may not even be possible, but I hope I've convinced the reader that either way, attempting a proof is not just a matter of mimicking some empirical observation. It will push mathematics to a new heights and force humanity to re-confront the very essence of what computability truly is.

Benjamin Kaminsky, B.A. is a graduate of the University of Miami.

Disclaimers

Author's Note on Naming Conventions

It is my explicit wish that no theorems in this work be named after me, nor any past ones retroactively modified. Any naming attached to objects (e.g., the KTH and KL) is intended solely as a means of reference and attribution for constructible entities, not for formal results. The focus of this paper is not personal legacy, but the advancement of geometric computability and its 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 KTH 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 speculative aspects of Yang–Mills theory, CPT symmetry, and geometric type theory/category 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 are powerful tools, but only if one can ask the right questions—which, in the end, is what mathematics is all about.

Email — benkaminskymath@gmail.com

Alternate definition for FSS. This is still up for debate. While ideally more rigorous with the distinction between the tuple, this is being left open for the time.

Definition (Fluidic State System): A Fluidic State System is a 6-tuple:

$$\mathcal{F} = (M, \Phi, \mathcal{G}, \Delta, \mu, \rho)$$

Where:

M is a finite configuration space of local geometric states. For example, $M = \Sigma \times \Theta$ where Σ is chirality and Θ is orientation.
Φ is a set of local flow rules, governing propagation between coupled units. That is,

$$\Phi : M^k \to \mathcal{G}$$

where k is the size of the neighborhood, and $\mathcal{G}$ is the group of allowable morphisms.
$\mathcal{G}$ is a group or groupoid of geometric transformations, such as (c_1, c_2, d_1, d_2) acting on manifold units.
Δ is a state evolution function, defined as:

$$\Delta : M^k \times \mathcal{G} \to M$$

Mapping neighborhood state + action to new state.
μ is a masking or observation function,

$$\mu : M \to O$$

where O is the classical observable (e.g., $\{0, 1\}$ on the display layer).
ρ is a clocking or update regime, such as a sweep rule or asynchronous propagation mode.

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] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *LISP and Symbolic Computation*, 6:289–360, 1993.

[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] Matthias Felleisen. On the expressive power of programming languages. *Science of Computer Programming*, 17(1-3):35–75, 1991.

[14] M. Atiyah. Mathematics in the 20th century. *Bulletin of the London Mathematical Society*, 34:1–15, 2002.

[15] Joshua Meyers. Computation and category theory. Topos Institute Blog.

[16] Paul Benacerraf. What numbers could not be. *The Philosophical Review*, 74(1):47–73, Jan 1965.

[17] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[18] 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.

[19] Terence Tao. On the universality of the incompressible Euler equation on compact manifolds. *Discrete and Continuous Dynamical Systems*, 38(3):1553–1565, 2018.

[20] brusspup. Amazing water and sound experiment #2. YouTube, 2013. Accessed: 2025-07-01.

[21] 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.

[22] 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.

[23] Peter M. Kogge. *The Architecture of Symbolic Computers*. McGraw-Hill, New York, 1991.

[24] G.J. Sussman, J. Holloway, G.L. Steel, and A. Bell. Scheme-79 — lisp on a chip. *Computer*, 14(7):10–21, 1981.

[25] R. C. Penner. The decorated Teichmüller space of punctured surfaces. *Communications in Mathematical Physics*, 113:299–339, June 1987.

[26] Brendan Fong. Decorated cospans, 2015.

[27] John C. Baez, Kenny Courser, and Christina Vasilakopoulou. Structured versus decorated cospans. *Compositionality*, 4:3, September 2022.

[28] Daniel Müllner. Orientation reversal of manifolds. *Algebraic and Geometric Topology*, 9(4):2361–2390, November 2009.

[29] Nadia Chouaieb, Alain Goriely, and John H. Maddocks. Helices. *Proceedings of the National Academy of Sciences*, 103(25):9398–9403, 2006.

[30] Toshiaki Adachi. Foliation on the moduli space of helices on a real space form. *International Mathematical Forum*, 4(35):1699–1707, 2009.

[31] Gino Loria. *Spezielle algebraische und transscendente ebene Kurven. Theorie und Geschichte.* B. G. Teubner, Leipzig, 1902.

[32] László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.

[33] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *CoRR*, abs/1812.01537, 2018.

[34] Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Computer Science: Faculty Publications, Smith College*, 2005.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] Jacob Lurie. On the classification of topological field theories. *Current Developments in Mathematics*, 2008:129–280, 2009.

[41] Ulrik Buchholtz and Johannes Schipp von Branitz. Primitive recursive dependent type theory, 2024.

[42] S. K. Donaldson. An application of gauge theory to the topology of 4-manifolds. *Journal of Differential Geometry*, 18:279–315, 1983.

[43] Andreas Blass. Topoi and computation. Written version of a talk given at the C.I.R.M. Workshop on Logic in Computer Science, June 1988. Mathematics Dept., University of Michigan.

[44] Kenichi Morita. *Theory of reversible computing*. Springer, 2017.

[45] 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.

[46] Tibor Radó. On non-computable functions. *The Bell System Technical Journal*, 41(3):877–884, May 1962.

[47] Ángel González-Prieto, Eva Miranda, and Daniel Peralta-Salas. Topological Kleene Field Theories: A new model of computation. *arXiv preprint arXiv:2503.16100*, 2025.

[48] Maxim Kontsevich. Deformation quantization of poisson manifolds, I. *Letters in Mathematical Physics*, 66(3):157–216, 2003.

[49] Alain Connes. Non-commutative geometry and the Standard Model of elementary particle physics. *Journal of High Energy Physics*, 2000.

[50] Edward Witten. Quantum field theory and the Jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, 1989.

[51] A. Yu. Kitaev. Unpaired Majorana fermions in quantum wires. *Physics-Uspekhi*, 44(10S):29–36, 2001. Translated from Uspekhi Fizicheskikh Nauk, Volume 171, Issue 1, 2001.

[52] Konstantin Y. Bliokh, Daniel Leykam, Max Lein, and Franco Nori. Topological non-Hermitian origin of surface Maxwell waves. *Nature Communications*, 10, 2019.

[53] Jacob A. Barandes. The stochastic-quantum correspondence, 2023.

[54] 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.

[55] T. C. Kraan and P. van Baal. Periodic instantons with non-trivial Holonomy. *Nuclear Physics B*, 533(1-3):627–659, 1998.

[56] F. R. Klinkhamer and N. S. Manton. A saddle-point solution in the Weinberg-Salam theory. *Physical Review D*, 30(10):2212–2220, 1984.

[57] G. 't Hooft. A property of electric and magnetic flux in nonabelian gauge theories. *Nuclear Physics B*, 153:141–160, 1979.

[58] E. Witten. Topological Quantum Field Theory. *Communications in Mathematical Physics*, 117(3):353–386, 1988.

[59] S. Coleman. Nonabelian plane waves. *Physics Letters B*, 70(1):59–60, 1977.

[60] 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.

[61] S. K. Donaldson. Yang-mills theory and geometry. Unpublished article, Imperial College, London, January 2005.

[62] H. A. Kastrup. Canonical theories of lagrangian dynamical systems in physics. *Physics Reports*, 101(1-2):151, 1983.

[63] 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.

[64] Int'l Centre for Theoretical Physics. Conversation: Salam, sciama, witten and budinich, Oct 2014. Accessed: 2025-06-18.

[65] Merlin Carl. The lost melody theorem for infinite time blum-shub-smale machines, 2020.

[66] Joel David Hamkins and Andy Lewis. Infinite time turing machines, 1998.

[67] Toby Cubitt, David Perez-Garcia, and Michael M. Wolf. Undecidability of the spectral gap. *Forum of Mathematics, Pi*, 10, 2022.

[68] Matthew P. Szudzik. *The Computable Universe Hypothesis*, page 479–523. WORLD SCIENTIFIC, October 2012.

[69] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47(10):777–780, 1935.