Computing with Spherepop: Foundations of Geometric Merge–Collapse Computation

Flyxion

December 8, 2025

Abstract

Spherepop is a model of computation in which computational values are represented by geometric regions and in which the evolution of a program is governed by two primitive operations: merge and collapse. The merge operation combines regions through spatial union, while collapse abstracts internal detail by projecting regions into canonical forms. This formulation yields a geometric computational calculus in which spatial interaction, abstraction, and composition replace symbolic manipulation. The purpose of this monograph is to develop Spherepop as a formal computational framework, to demonstrate its theoretical completeness, and to establish foundational connections between geometric computation, neural computation, tensor logic, and differential learning. Particular attention is given to the formal semantics of merge and collapse, the expressive power of merge–collapse calculi, the complexity theory of geometric evaluation, and the categorical structure underlying Spherepop programs. This monograph also investigates differentiable variants, formal relations with logical calculi and spatial computing models, and empirical applications to machine learning. Throughout, emphasis is placed on rigorous mathematical treatment, theoretical generality, and formal foundations of geometric computation.

Contents

1 Introduction

Computational systems are traditionally formulated in symbolic or algebraic terms. Lambda calculus, Turing machines, register machines, and functional programming languages all describe computation as symbolic transformation. Modern machine learning retains this emphasis, expressing neural networks as compositions of matrix transformations and nonlinearities. In each case, computation is represented through algebraic structures, and semantics are specified by symbolic substitution or functional composition.

Spherepop proposes an inversion of this paradigm. Rather than representing computational values as symbols, expressions, or numerical arrays, Spherepop represents values as geometric regions embedded in an ambient space. Computation proceeds not by symbolic substitution but by spatial transformation: regions interact, merge, and collapse. From this perspective, computation becomes a geometric process of aggregation and abstraction.

Merge corresponds to spatial combination, while collapse corresponds to removal of local detail and projection into equivalence classes.

The central purpose of this monograph is to develop Spherepop as a formal model of computation. This requires precise mathematical definitions of regions, merge, collapse, evaluation, and equivalence. The intended theoretical contributions are fourfold. First, to establish Spherepop as a computational calculus with well-defined semantics. Second, to prove that Spherepop possesses expressive power equivalent to universal models, including lambda calculus. Third, to articulate a complexity theory characterizing merge–collapse evaluation. And fourth, to present Spherepop as a geometric foundation for machine learning and differentiable computation, including connections to tensor logic and neural representation.

Spherepop is not merely a geometric analogy for symbolic computation; rather, it is a computational formalism whose primitives are geometric. Geometric computation in the sense developed here should not be confused with geometric deep learning, geometric programming, or spatial computing architectures, although conceptual relations exist. The emphasis of Spherepop is foundational: it aims to provide a computational language whose semantics are spatial from the outset. Algebra, logic, and machine learning are regarded as specializations or projections of geometric computation into algebraic domains.

The scope of this monograph therefore extends beyond introductory exposition. The following pages develop formal semantics, computational expressiveness, operational equivalence, complexity analysis, differentiable variants, logical encodings, categorical interpretations, empirical implementations, and educational applications. The treatment is deliberately systematic, with definitions, theorems, and proofs supplied where appropriate. Many concepts introduced informally in earlier expository texts are rigorously developed here.

2 Mathematical Preliminaries

The formal development of Spherepop rests on a geometric representation of computational values. In order to establish well-defined semantics, we begin by rigorously specifying the notion of a geometric region, the ambient topological space in which computations take place, and the structural properties that merge and collapse are assumed to satisfy. The aim of this section is not to introduce Spherepop itself, but to lay the groundwork for its definition and analysis.

2.1 Ambient Space

Throughout this monograph we take the ambient space to be a Euclidean domain $\mathbb{R}^d$ for a fixed dimension $d \geq 1$. While all constructions are presented in the Euclidean setting, most results generalize to metric spaces and, with additional hypotheses, to general Hausdorff spaces. For conceptual clarity and geometric intuition, we restrict to Euclidean spaces unless explicitly stated otherwise. All regions are subsets of $\mathbb{R}^d$ endowed with the standard Euclidean topology.

2.2 Regions

We now specify the regularity conditions that regions are required to satisfy. A minimal set of assumptions is necessary both for geometric interpretation and for well-defined merge and collapse operations. In particular we require connectedness, boundedness, and measurability, so that collapse may be defined as a projection into a family of canonical representatives.

A *region* in $\mathbb{R}^d$ is a bounded, connected, Lebesgue-measurable subset $A \subseteq \mathbb{R}^d$ with nonempty interior.

Connectedness ensures that a region behaves as a single computational value rather than a multivalued collection. Boundedness provides finiteness in both measure and spatial extent, preventing pathological collapse behavior. Lebesgue-measurability allows collapse to be defined through integration, measure contraction, or other analytic constructions. Nonempty interior avoids degenerate cases such as isolated points and finite sets that do not meaningfully represent extended geometric values.

2.3 Topological and Measure-Theoretic Assumptions

Although regions may be arbitrary measurable sets, we shall frequently restrict attention to compact regions with piecewise smooth boundaries. This assumption is not strictly necessary for the formal development, but it simplifies proofs concerning collapse regularity and stability. Where additional regularity is required, it will be stated explicitly.

It will be convenient to denote by $\mathcal{R}_d$ the class of all admissible regions in $\mathbb{R}^d$. All Spherepop values belong to $\mathcal{R}_d$. Unless otherwise noted, set-theoretic operations are interpreted as operations in $\mathcal{R}_d$ when they preserve admissibility; otherwise collapse is implicitly applied to restore admissibility.

2.4 Equivalence of Regions

Collapse is intended to abstract internal structure while preserving global identity. This motivates a quotient construction: distinct regions may represent the same collapsed value. To capture this idea formally, we introduce an equivalence relation.

Two regions $A, B \in \mathcal{R}_d$ are *equivalent* if they belong to the same collapse class, written $A \sim B$, if $\text{pop}(A) = \text{pop}(B)$.

The collapse operator will be introduced formally in Section 3. For now we treat $\sim$ as an abstract equivalence relation satisfying reflexivity, symmetry, and transitivity. In practice, $\sim$ identifies regions whose interior structure is computationally irrelevant and whose global geometric representation is equivalent under collapse.

The quotient space $\mathcal{R}_d/\sim$ forms the semantic domain of Spherepop values. Computation acts on equivalence classes rather than raw regions.

2.5 Measurable Union and Regularization

The merge operation is defined at the level of raw regions by topological union. Since unions of admissible regions need not remain admissible (e.g. unions may lose connectedness or interior), merge is followed by collapse to enforce admissibility and canonicalization. In order to formalize this process, we introduce a standard regularization operation.

For any measurable set $S \subseteq \mathbb{R}^d$ with positive measure, the *regularization* $\text{reg}(S)$ is the unique connected component of the closure of the interior of S having maximal Lebesgue measure.

Regularization provides a canonical choice of connected representative for arbitrary unions. Although this construction is not itself collapse, it permits clean formulation of merge without reference to collapse. Collapse will be introduced as a projection from admissible sets to equivalence classes.

2.6 Summary of Structural Assumptions

The assumptions of boundedness, connectedness, and measurability define a class of geometric values amenable to collapse. The equivalence relation $\sim$ identifies collapsed values, while regularization ensures that merge acts on connected representatives of unions. These preliminaries are minimal in the sense that all later constructions rely only on these regularity conditions. Stronger smoothness assumptions may be introduced when needed for differentiable variants, but are not required for basic Spherepop semantics.

3 Collapse, Merge, and Canonicalization

We now provide formal definitions for the two primitive operations underlying Spherepop computation. The collapse operation takes an admissible region and returns a canonical representative of its equivalence class. The merge operation is defined compositionally as the union of regions followed by collapse. These definitions establish the semantic core of Spherepop and support the algebraic, topological, and computational arguments developed in subsequent sections.

3.1 Collapse as Canonical Projection

The collapse operator, denoted by pop, is intended to discard interior detail of a region and to assign a canonical geometric representative drawn from a family of normalized shapes. In order to define collapse formally, we require that admissible regions admit a unique canonical representative under an appropriate equivalence relation. We begin by assuming the existence of a projection map into $\mathcal{R}_d$.

A *collapse operator* is a mapping

$$\mathrm{pop} : \mathcal{R}_d \longrightarrow \mathcal{R}_d$$

satisfying the following axioms:

1. $\mathrm{pop}(A)$ is equivalent to A, that is $\mathrm{pop}(A) \sim A$;
2. idempotence: $\mathrm{pop}(\mathrm{pop}(A)) = \mathrm{pop}(A)$ for all $A \in \mathcal{R}_d$;
3. stability: if $A \sim B$, then $\mathrm{pop}(A) = \mathrm{pop}(B)$.

The idempotence condition expresses the fact that collapse is a projection onto canonical forms. Equivalence stability ensures that collapse acts on equivalence classes, thereby permitting denotational interpretation in the quotient space $\mathcal{R}_d/\sim$. Although we do not commit to a specific collapse rule, the canonical choices in practice include minimal enclosing regions, convex hulls, spherical approximations, or measure-contraction operators. These choices give rise to models of computation having different geometric regularities but identical formal structure.

3.2 Canonical Representations

Each equivalence class under $\sim$ contains infinitely many geometric representatives. Collapse selects a unique canonical representative, thereby enabling semantic comparison of Spherepop values. This yields the following proposition, whose proof is immediate from the axioms of collapse.

For each $A \in \mathcal{R}_d$, the element $\mathrm{pop}(A)$ is the unique canonical representative of the equivalence class of A under $\sim$.

Canonical representatives support unambiguous evaluation of merge and collapse expressions by erasing interior structure. This characteristic is essential for formal semantics: without canonicalization, the semantics would depend on geometric details irrelevant to computation.

3.3 Merge as Union Followed by Collapse

Merge is intended to combine interacting regions and then apply collapse to remove redundant internal distinctions. Since raw union may fail to satisfy admissibility conditions, merge is defined as a composition of union and collapse. Using regularization to maintain connectedness, we obtain the following definition.

For regions $A, B \in \mathcal{R}_d$, the *merge* of A and B is

$$A \diamond B := \mathrm{pop}\Big(\mathrm{reg}(A \cup B)\Big).$$

The regularization step ensures that the argument of collapse lies in $\mathcal{R}_d$. Collapse then projects the regularized union into the canonical representative. Computationally, merge produces a new value reflecting the interactive geometry of A and B.

3.4 Idempotence and Equivalence Preservation

The algebraic properties of merge follow from those of collapse. Since $\mathrm{reg}(A \cup A) = A$ and $\mathrm{pop}(A)$ is canonical, we obtain the following lemma.

Merge is idempotent in the sense that $A \diamond A = \mathrm{pop}(A)$ for all $A \in \mathcal{R}_d$.

Equivalence is preserved under merge: if $A \sim A'$ and $B \sim B'$, then $\mathrm{reg}(A \cup B) \sim \mathrm{reg}(A' \cup B')$, hence collapse assigns the same canonical representative. The following result formalizes this statement.

Merge respects equivalence, that is, if $A \sim A'$ and $B \sim B'$, then $A \diamond B = A' \diamond B'$.

3.5 Associativity up to Collapse

In contrast to idempotence, associativity requires greater care. Union of sets is associative, but regularization and collapse may interfere with naive associativity. Nevertheless, merge is associative up to equivalence and thus associative in the quotient domain of canonical forms. The precise formulation is as follows.

Merge is associative up to collapse. More precisely,

$$(A \diamond B) \diamond C \;=\; \mathrm{pop}(A \cup B \cup C) \;=\; A \diamond (B \diamond C).$$

Consequently, merge is associative in the quotient space $\mathcal{R}_d/\sim$.

Proof. The equivalence follows from associativity of union and idempotence of collapse. A formal proof is obtained by applying regularization and collapse successively to both sides and invoking uniqueness of canonical representatives. Details are routine and omitted. □

The associativity of merge is crucial for the algebraic structure developed in later sections. It permits the definition of composite merge expressions without parentheses and supports the categorical interpretations in Section 8.

3.6 Semantic Consequences

The axioms introduced above imply that Spherepop expressions denote canonical representatives in $\mathcal{R}_d$. Merge and collapse behave as projection-based algebraic operations. The equivalence relation $\sim$ captures semantic identity of geometric values. In subsequent sections, we develop Spherepop as a computational calculus built from these primitives. The remainder of the monograph derives semantic interpretations, operational evaluation rules, categorical structures, and expressiveness results from these definitions.

4 Syntax and Semantics of Spherepop Programs

Having established collapse and merge at the level of geometric values, we now introduce Spherepop programs as syntactic expressions built from these primitives. The aim of this section is to specify the syntax of Spherepop, the operational semantics governing evaluation, and the denotational semantics that interprets programs as canonical geometric values. These foundations are essential for subsequent discussions of expressiveness, complexity, and equivalence with classical computational systems.

4.1 Syntax

Spherepop terms represent formal expressions over merge and collapse. The syntax resembles functional calculi, although the primitives operate on geometric values. We introduce the abstract syntax as follows.

The class of *Spherepop terms* is the smallest set generated by the grammar

$$t ::= A \mid \mathrm{pop}(t) \mid t_1 \diamond t_2,$$

where A is an atomic symbol representing a primitive region.

The atomic symbols constitute an alphabet of primitive geometric values, typically regarded as parameters of the program. From this alphabet, more complex expressions are formed by applying the collapse operator and by combining expressions via merge. In contrast to lambda calculus or term-rewriting systems, no variables, abstractions, or function applications appear in the syntax; composition occurs solely through merge.

4.2 Operational Semantics

Operational semantics describes how Spherepop terms evaluate. Since merge and collapse have already been defined at the level of geometric values, we lift these operations to the syntactic domain by interpreting terms during evaluation. We write t for the geometric region denoted by t. Evaluation is defined inductively by the following clauses:

$$A = A, \qquad \mathrm{pop}(t) = \mathrm{pop}(t), \qquad t_1 \diamond t_2 = \mathrm{pop}\Big(\mathrm{reg}(t_1 \cup t_2)\Big).$$

Evaluation is viewed as a reduction process in which collapse is applied whenever it is present syntactically, and merge reduces adjacent terms by collapsing their union. The following definition provides an operational reduction relation.

The *Spherepop reduction relation* $\to$ is the smallest relation on terms such that

$$\mathrm{pop}(A) \to \mathrm{pop}(A), \qquad \mathrm{pop}(\mathrm{pop}(t)) \to \mathrm{pop}(t),$$

$$(t_1 \diamond t_2) \diamond t_3 \to t_1 \diamond (t_2 \diamond t_3), \qquad \mathrm{pop}(t_1 \diamond t_2) \to \mathrm{pop}(\mathrm{pop}(t_1) \cup \mathrm{pop}(t_2)).$$

These rules formalize collapse absorption and merge associativity at the syntactic level. Although evaluation reduces to canonical representatives, the operational calculus admits multiple syntactic reduction sequences. The following proposition asserts that evaluation terminates under mild assumptions.

Every Spherepop term reduces, in a finite number of steps, to a normal form of the shape pop(A) for some region A.

The proof uses induction on term structure together with idempotence of collapse and associativity of merge. Details will be provided in Section 6, where termination and confluence are examined in full generality.

4.3 Normal Forms

Spherepop evaluation produces canonical geometric values. We therefore define normal forms syntactically by identifying collapse expressions that cannot be further reduced.

A Spherepop term t is in *normal form* if $\mathrm{pop}(t) = t$.

Equivalently, normal forms are terms of the shape pop(A) containing no interior collapse expressions. Every Spherepop term reduces to a unique normal form in the quotient space of canonical representatives. The following result is immediate from idempotence of collapse and associativity of merge.

Every Spherepop term t has a unique normal form up to equivalence under $\sim$.

Uniqueness holds because collapse selects a canonical representative of the equivalence class of the union of all primitive regions appearing in t. In fact, the geometry of reduction is strictly determined by the placement of merge and collapse operations, but the ultimate result is canonical.

4.4 Denotational Semantics

While operational semantics describes evaluation, denotational semantics assigns a meaning to each term independent of reduction. The semantic domain of Spherepop is the quotient space $\mathcal{R}_d/\sim$, whose elements are equivalence classes of admissible regions.

The *denotational semantics* of a term t is

$$t_\sim := [t]_\sim \in \mathcal{R}_d/\sim,$$

the equivalence class of the geometric region computed by evaluating t. Since collapse is idempotent and respects equivalence, denotational semantics is well-defined.

Denotational semantics is sound with respect to operational reduction: if $t \to t'$, then $t_\sim = t'_\sim$.

Proof. Reduction either collapses a term or associates a merge expression; both operations preserve canonical equivalence by the axioms of merge and collapse. Thus evaluation preserves denotational interpretation. $\square$

4.5 Summary

Spherepop terms denote canonical geometric representatives in the quotient space $\mathcal{R}_d / \sim$. Operational reduction corresponds to computing these representatives syntactically, while denotational semantics interprets them directly through projection. The existence of unique normal forms establishes correctness of evaluation. These foundations allow us to examine expressiveness, equational reasoning, operational equivalence, and complexity, all of which rely on the interaction of merge and collapse in the quotient domain.

5 Expressiveness and Computational Universality

We now address the fundamental question of expressive power. Spherepop was introduced as a geometric computational calculus, but no geometric description alone guarantees universal computability. In this section we demonstrate that Spherepop possesses the same expressive capacity as classical models of computation. More precisely, we show that Spherepop can simulate, up to canonical equivalence, Turing-complete calculi. The proofs proceed by encoding -calculus into Spherepop, and by establishing that merge–collapse evaluation reproduces -reduction under a geometric interpretation. This establishes Spherepop as a universal computational formalism.

5.1 Encoding Boolean Values

Boolean values are represented geometrically by selecting two canonical regions. The particular choice is irrelevant modulo collapse. For concreteness, let T and F be two primitive regions belonging to distinct equivalence classes under $\sim$. Boolean connectives are represented by merge–collapse constructions that correspond to logical operations through canonical projection. For example, logical conjunction is represented by

$$\mathsf{and}(A, B) = A \diamond B.$$

Disjunction and negation are handled analogously by selecting canonical representatives for their collapse classes. These encodings respect logical identities and are sufficient to express boolean computation. That boolean operations admit geometric encodings is the first indication of Spherepop's expressive capacity but does not yet imply universality.

5.2 Encoding Function Application

Spherepop does not include variables, binding, or substitution syntactically. These structures must therefore be encoded geometrically. The key observation is that collapse erases interior structure, providing a geometric analogue of abstraction. Merge combines geometric values so that subsequent collapse yields the analogue of -reduction.

The challenge of encoding -abstraction is to represent a function as a region whose geometric structure depends parametrically on its argument. To express this, we introduce parametric regions generated by pairing atomic symbols with formal placeholders. Given an atomic symbol A, we define a parametric region $A[x]$ which acts as a geometric proxy for a variable occurrence. The collapse operator projects $A[x]$ to a canonical form whenever x is instantiated by a concrete region. Function application corresponds to merge, while abstraction is represented by canonical collapse eliminating the placeholder.

5.3 Spherepop Simulation of λ-Calculus

We now outline the encoding of -terms. Let Λ denote the set of -terms generated by variables, abstraction, and application:

$$M ::= x \mid \lambda x.M \mid M\,N.$$

To each -term M we associate a Spherepop term $\langle M\rangle$ defined inductively. Variables are mapped to parametric regions, abstraction discards the parameter via collapse, and application corresponds to merge:

$$\langle x\rangle = A[x], \qquad \langle \lambda x.M\rangle = \mathrm{pop}(\langle M\rangle), \qquad \langle M\,N\rangle = \langle M\rangle \diamond \langle N\rangle.$$

Application of $\diamond$ corresponds to function application, while the collapse in $\langle \lambda x.M\rangle$ corresponds to abstraction eliminating the placeholder x. Under this encoding, -reduction becomes equivalent to merge followed by collapse, so that the -term $(\lambda x.M)\,N$ reduces geometrically to

$$\mathrm{pop}\Big(\mathrm{reg}(\langle M\rangle \cup \langle N\rangle)\Big),$$

which is equal to $\langle M[N/x]\rangle$ in the quotient space.

5.4 Correctness of the Encoding

The correctness of the encoding is expressed by soundness and completeness with respect to -reduction. Soundness asserts that if $M \to_\beta N$, then $\langle M\rangle \to \langle N\rangle$ in Spherepop. Completeness asserts that if $\langle M\rangle \to \langle N\rangle$, then $M \to_\beta N$. These results ensure that Spherepop faithfully simulates -calculus evaluation.

[Soundness] If $M \to_\beta N$ in -calculus, then $\langle M\rangle \to \langle N\rangle$ in Spherepop.

Proof. -reduction eliminates variable occurrence in application of abstractions. Under the encoding, this corresponds to a merge followed by collapse eliminating the placeholder. Since collapse removes parametric structure and merge preserves equivalence, reduction in -calculus produces the same canonical representative as Spherepop evaluation. A full proof is given in Appendix A. □

[Completeness] If $\langle M\rangle \to \langle N\rangle$ in Spherepop, then $M \to_\beta N$ in -calculus.

Proof. Merge may only combine parametric and non-parametric regions that represent syntactic application. Collapse may only eliminate placeholders corresponding to bound variables. It follows that every reduction step in the Spherepop encoding corresponds to a -reduction in the source -term. Detailed argumentation is supplied in Appendix A. □

5.5 Turing-Completeness of Spherepop

We are now in a position to state the main result of this section. Since -calculus is Turing complete, and since Spherepop simulates -calculus with soundness and completeness, it follows that Spherepop itself is Turing complete.

[Computational Universality] Spherepop is a Turing-complete computational formalism.

Proof. Immediate from soundness and completeness with respect to -calculus. Since -calculus is Turing complete, Spherepop can express every computable function under the merge–collapse encoding. The details follow from Theorems 5.1 and 5.2. □

5.6 Expressive Significance

Spherepop does not merely simulate symbolic computation; it provides a geometric formalism in which computation is accomplished through merge and collapse rather than symbolic substitution. The equivalence with -calculus shows that the absence of symbolic syntax does not diminish expressive power. Instead, geometric interaction suffices to encode universal computation. The remainder of this monograph explores the consequences of this fact for operational semantics, complexity, categorical structure, and differentiable computation.

6 Termination, Confluence, and Normalization

Having established expressiveness and completeness, we now examine the dynamics of Spherepop evaluation from the perspective of rewriting theory. The merge–collapse calculus defines a rewriting system on syntactic terms. The purpose of the present section is to investigate three fundamental properties: termination, confluence, and the structure of normal forms. We show that, under the collapse axioms of Section 3, Spherepop evaluation terminates, is confluent up to equivalence, and yields unique canonical normal forms in the quotient space of regions modulo $\sim$.

6.1 Termination

Termination asserts that every evaluation sequence eventually reaches a normal form. Spherepop terms are finite, collapse is idempotent, and merge is associative up to collapse; these properties ensure that evaluation does not produce infinite reduction chains. The proof proceeds by structural induction on terms and by observing that every reduction decreases a well-founded measure associated with the syntactic complexity of expressions.

[Termination] Every Spherepop term reduces to a normal form in a finite number of steps.

Proof. Define the complexity of a term t, denoted $|t|$, to be the number of merge and collapse symbols occurring in t. Every reduction either removes a redundant collapse, associates a merge, or reduces an interior collapse occurrence. In each case, the resulting term strictly reduces in $|t|$. Since $|t|$ is a natural number, repeated reduction must terminate. Details are given in Appendix B. □

Termination confirms that Spherepop evaluation always reaches a canonical collapsed form and therefore produces a well-defined denotational value. Together with uniqueness of canonical representatives, this provides a strong semantic foundation for the calculus.

6.2 Confluence

Confluence asserts that evaluation order does not affect the ultimate result. More precisely, if a term admits two different reduction sequences, then both sequences must lead to equivalent canonical representatives. Although merge may be applied in different orders, associativity up to collapse guarantees that the resulting canonical representative is invariant under reduction path. The result is analogous to the Church–Rosser property of λ-calculus.

[Confluence up to equivalence] If $t \to^* u$ and $t \to^* v$, then $u \to^* w$ and $v \to^* w'$ for some normal forms w, w' satisfying $w \sim w'$.

Proof. Evaluation of t produces a canonical representative of the union of all primitive regions occurring in t. Regardless of the reduction path, merge and collapse eliminate syntactic variation and produce equivalent canonical representatives. A formal proof is obtained by adapting the standard parallel reduction argument for λ-calculus and using associativity up to collapse rather than strict associativity. Complete details appear in Appendix B. $\square$

Confluence up to equivalence suffices for semantic coherence: denotational semantics assigns a unique canonical form to each term even if syntactic evaluation proceeds along different paths.

6.3 Normalization

Since reduction terminates and is confluent up to equivalence, every Spherepop term has a unique canonical normal form. The canonical normal form corresponds to the collapse of the regularized union of all primitive regions appearing syntactically in the term. In particular, the normal form is independent of the order in which merges and collapses are evaluated.

[Normalization] Every Spherepop term has a unique normal form, up to equivalence under $\sim$.

Proof. By termination and confluence up to equivalence, the standard normalization argument applies. Each reduction sequence reaches a normal form. Since all such normal forms are equivalent under collapse, the canonical representative is unique in the quotient $\mathcal{R}_d/\sim$. This representative is the denotational meaning of the term. $\square$

Normalization shows that Spherepop is a strongly normalizing rewriting system with unique canonical solutions modulo $\sim$. This property is essential for soundness of denotational semantics and for the categorical interpretations developed in Section 8.

6.4 Canonical Normal Forms

We now characterize normal forms geometrically. Let t be a Spherepop term and let $A_1, \ldots, A_n$ be the primitive regions appearing as atomic symbols in t. Then evaluation of t produces the canonical representative of the regularized union $A_1 \cup \cdots \cup A_n$. Collapse eliminates interior detail and provides a canonical representative of this union. Consequently, the semantic content of a Spherepop term is completely determined by the set of atomic regions occurring in the expression, modulo canonicalization.

The structure of canonical normal forms implies that the syntactic shape of a term determines its denotational value only through its primitive constituents, not through the arrangement of merges or collapses. In this respect, Spherepop resembles idempotent commutative semirings and certain classes of idempotent algebraic structures, although the geometry of regions introduces significantly richer semantics.

6.5 Consequences

Termination, confluence up to equivalence, and normalization establish Spherepop as a deterministic computational calculus at the level of canonical representatives. The denotational semantics introduced in Section 4 agrees with operational evaluation, providing a foundation for equational reasoning, expressiveness results, and categorical structure. These properties also facilitate the definition of differentiable Spherepop operators and the incorporation of merge–collapse into machine learning systems.

7 Complexity, Decidability, and Bounds on Evaluation

We now examine Spherepop from the perspective of computational complexity. Merge and collapse are geometric operations, and their evaluation depends on geometric regularity, metric dimensionality, topological properties, and the choice of collapse function. Our goal in this section is not to fix a particular implementation or representation, but to characterize the complexity of Spherepop evaluation abstractly, in relation to standard computational models. We also address decidability of evaluation properties, termination questions under extensions of the calculus, and comparative complexity with tensor-based computation.

7.1 Complexity of Merge and Collapse

Merge is defined by union followed by regularization and collapse. The complexity of evaluating a merge operation therefore depends on the complexity of computing regularized unions and canonical representatives. When regions are represented by analytic descriptions or algebraic surfaces, merge may be computed symbolically by combining equations. When regions are represented discretely, merge may be computed via geometric data structures and spatial algorithms. Although the complexity of these operations depends on representation, we identify uniform upper bounds in terms of region complexity and syntactic size of expressions.

Let $|t|$ denote the syntactic size of a Spherepop term and let $c(A)$ denote the geometric complexity of a region A under a fixed representation. Then the worst-case complexity of merge and collapse can be bounded by a function of $|t|$ and $\max c(A)$. For instance, if regions are represented by polygonal meshes, then merge can be computed in time polynomial in the number of faces. If regions are represented by implicit level sets, merge reduces to level-set operations whose complexity is polynomial in the resolution of the discretization.

Let t_1, t_2 be Spherepop terms with geometric representatives A_1, A_2. Under any representation whose merge and collapse operations run in time polynomial in the size of the representations of A_1 and A_2, the evaluation of $t_1 \diamond t_2$ is polynomial in the sizes of A_1 and A_2.

The proposition expresses a general complexity upper bound for merge–collapse operations, but the bound is abstract and depends on modeling assumptions. Later sections provide detailed complexity bounds for specific representations such as spherical approximations, voxel discretizations, and distance-field representations.

7.2 Decidability of Evaluation

Since Spherepop is Turing complete, many semantic questions concerning evaluation are undecidable in general. For instance, the question of whether two Spherepop expressions evaluate to equivalent canonical forms is undecidable. This follows directly from the undecidability of program equivalence in -calculus. Similarly, the halting problem for Spherepop programs is undecidable, as termination of evaluation is equivalent to termination of -calculus reduction under the encodings of Section 5. Nevertheless, certain properties become decidable under restrictions on region complexity or collapse operators.

The problems of program equivalence, halting, and normalization are undecidable for Spherepop.

Proof. Directly from Turing completeness. The encoding of -calculus into Spherepop is sound and complete, so any undecidable property of -calculus yields an undecidable property of Spherepop. □

Decidability results may be obtained for fragments of the calculus in which collapse is restricted to convex representatives or minimal enclosing balls; these fragments behave like geometric decision procedures and admit algorithmic evaluation. Such fragments are of interest for differentiable computation and low-dimensional neural approximations, but do not capture full expressiveness.

7.3 Upper and Lower Bounds

The worst-case complexity of Spherepop evaluation depends on the complexity of collapse. Under minimal enclosure collapse, complexity reduces to computing minimal enclosing convex bodies, which is polynomial in region complexity. Under convex hull collapse, complexity is bounded by the complexity of convex hull construction, which is polynomial in d and in the number of boundary facets. Under spherical collapse, complexity reduces to computing minimal enclosing spheres, which is linear or near-linear in ambient dimension depending on representation.

Lower bounds follow from the equivalence with -calculus. Since Spherepop simulates -calculus, evaluation of Spherepop expressions can require exponential time in $|t|$ in the worst case. Indeed, the classical exponential lower bounds for certain -expressions carry over directly to Spherepop under the encoding of Section 5.

7.4 Comparative Complexity

Comparisons between Spherepop and tensor-based computation depend on representation. Under discrete representations, merge corresponds to concatenation and collapse corresponds to pooling or projection. Under differentiable representations, collapse corresponds to a differentiable projection operator akin to attention pooling or normalization. In such settings, Spherepop evaluation admits complexity comparable to standard neural architectures.

In representation-agnostic terms, Spherepop is at least as complex as -calculus evaluation, and at least as complex as tensor contraction under standard embeddings. These statements reflect expressiveness but do not preclude practical efficiency when using appropriate geometric representations. Indeed, when collapse is implemented via spherical approximation or minimal enclosing volume, merge–collapse pipelines can run with complexity comparable to convolutional neural networks and pooling operations.

7.5 Consequences

Spherepop is computationally universal, and therefore inherits undecidability of program equivalence and halting. Nevertheless, when restricted to particular geometric representatives, merge–collapse evaluation admits efficient implementations and polynomial complexity bounds. The abstract complexity of Spherepop evaluation therefore depends jointly on syntactic structure and geometric representation. These results motivate the differentiable implementations considered in Section 9 and the machine-learning applications developed in Section 10.

8 Algebraic and Categorical Structure

We have thus far developed Spherepop as a geometric calculus whose primitive operations are merge and collapse. In this section we show that these operations carry a natural algebraic and categorical structure. In particular, merge induces an idempotent commutative monoid structure on canonical representatives, collapse induces a quotient identifying idempotent projections, and the interplay between the two supports a symmetric monoidal category structure. These structures will later permit comparison with algebraic calculi such as tensor logic and categorical models of computation.

8.1 Idempotent Semigroups and Canonical Representatives

Let $\mathcal{S}$ denote the set of canonical representatives $\mathrm{pop}(A)$ for $A \in \mathcal{R}_d$. Since every region is equivalent to its collapsed representative, and since merge is associative up to equivalence, we may regard $\diamond$ as an associative binary operation on $\mathcal{S}$. Idempotence follows immediately from $\mathrm{pop}(A) \diamond \mathrm{pop}(A) = \mathrm{pop}(A)$, and commutativity follows from commutativity of set union modulo canonicalization. We therefore obtain:

The set $\mathcal{S}$ equipped with merge forms a commutative idempotent semigroup.

This algebraic structure is reminiscent of idempotent semirings and tropical algebraic structures. The collapse operator acts as a projection onto $\mathcal{S}$, giving rise to a retraction from $\mathcal{R}_d$ into the semigroup $(\mathcal{S}, \diamond)$.

8.2 Monoidal Structure

Because merge is associative and symmetric up to canonical equivalence, the structure $(\mathcal{S}, \diamond)$ is a symmetric monoid. The identity element is obtained by collapsing the empty interior region, which serves as a neutral element under merge. We therefore obtain a symmetric monoidal structure on canonical representatives. Formally:

The canonical representative domain $\mathcal{S}$ with operation $\diamond$ is a symmetric monoidal category with a single object whose endomorphisms are canonical regions under merge–collapse equivalence.

Proof. A symmetric monoidal category with one object is equivalent to a commutative monoid. Associativity and commutativity up to canonical equivalence have already been established, and the neutral element is given by the collapsed empty region. Hence the claim follows. □

This categorical interpretation establishes Spherepop values as morphisms in a monoidal category, with merge as monoidal product. Collapse appears as a retraction from $\mathcal{R}_d$ into the monoidal category $\mathcal{S}$.

8.3 Tensorial Interpretation

The monoidal operation $\diamond$ provides a tensor-like operation on canonical representatives. Indeed, merge behaves analogously to tensor product in categorical semantics of computation, with collapse providing normalization and canonicalization of tensor expressions. In this sense, Spherepop may be understood as a geometric realization of algebraic tensor structures, in which monoidal composition is realized by geometric aggregation and idempotent projection.

Although Spherepop does not contain an internal notion of tensor as a multilinear algebraic object, the monoidal structure supports a tensorial interpretation of computation: merge corresponds to parallel composition, and collapse corresponds to elimination of redundant structure. From this perspective, Spherepop may be regarded as a geometric variant of tensor-based computation in which spatial interaction replaces algebraic multiplication.

8.4 Comparison with Tensor Logic

Tensor logic, in the sense articulated by Domingos, interprets logical inference and learning through tensorial composition, allowing logical operations to be reformulated as tensor operations. Spherepop provides an analogous structure in which merge corresponds to tensor-style composition and collapse corresponds to logical abstraction. The essential difference is that Spherepop grounds tensor operations in geometry rather than algebra. Logical combination becomes geometric union with normalization rather than tensor multiplication.

From the standpoint of semantics, tensor logic identifies truth values with tensor elements and logical operations with tensor algebra. Spherepop instead identifies computational values with geometric regions and computational operations with merge and collapse. The structural role of tensor product is therefore played by merge. Logical implication and abstraction are represented by collapse, which removes interior structure and produces canonical representatives. In this interpretation, the algebra of canonical regions under merge becomes a geometric tensor calculus with an idempotent projection operator.

8.5 Duality between Geometry and Tensorial Semantics

The monoidal interpretation of Spherepop expresses a duality between geometric and tensorial semantics. On one hand, merge corresponds to geometric aggregation; on the other, it corresponds to monoidal composition. Collapse corresponds to elimination of geometric fine structure; on the other, it corresponds to tensorial abstraction. In this dual interpretation, geometric interaction encodes algebraic combination, and collapsing geometric detail corresponds to eliminating algebraic redundancy. The calculus thereby supports a tensorial interpretation without relying upon algebraic tensor definitions; geometry itself realizes the tensor structure.

8.6 Consequences

The algebraic and categorical interpretation of Spherepop shows that merge and collapse induce a symmetric monoidal structure on canonical representatives. This structure is sufficiently expressive to provide a geometric foundation for tensor-style computation and geometric interpretations of tensor logic. From this perspective, tensor logic appears not as an external framework but as an algebraic manifestation of Spherepop's internal monoidal structure. The resulting algebraic semantics motivates the categorical interpretations and logical embeddings developed in the following sections.

9 Differential and Continuous Variants

The preceding development has treated collapse and merge as discrete geometric primitives. In practical computation, particularly in machine learning, continuous variants of these operations are essential. In this section we introduce continuous and differentiable formulations of merge and collapse, establishing a geometric counterpart to differentiable programming. These continuous formulations provide a path to incorporating gradient-based optimization, backpropagation, and learning while retaining the geometric perspective that distinguishes Spherepop from tensor-based models.

9.1 Continuous Collapse

Let $\mathcal{R}_d$ denote a suitable function space of regions equipped with a metric or topological structure. Rather than defining pop as a projection onto canonical representatives, we define a one-parameter family of continuous operators

$$\mathrm{pop}_\tau : \mathcal{R}_d \to \mathcal{R}_d, \qquad \tau \in [0, 1],$$

with $\mathrm{pop}_0 = \mathrm{id}$ and $\mathrm{pop}_1 = \mathrm{pop}$, such that pop_τ depends smoothly on τ. Intuitively, τ controls the degree of collapse, with $\tau = 0$ representing full geometric detail and $\tau = 1$ representing complete abstraction. In voxel, mesh, or implicit-surface implementations, pop_τ may be realized by progressive smoothing, erosion, or canonicalization of interior structure.

This continuous family induces a trajectory from a detailed region to its canonical representative,

$$A_\tau = \mathrm{pop}_\tau(A), \qquad \tau \in [0, 1],$$

providing a geometric analog of continuous relaxation and smoothing flows used in differentiable rendering and implicit neural representations.

9.2 Differentiable Merge

Merge is defined algebraically by $A \diamond B = \mathrm{pop}(A \cup B)$, but the geometric union and subsequent collapse can be replaced by differentiable approximations. Let U_ε denote a differentiable blending operator that approximates set union for small $\varepsilon > 0$,

$$A \,\tilde{\cup}_\varepsilon\, B = U_\varepsilon(A, B),$$

where U_ε is chosen from a family of smooth blending operators such as soft-union or differentiable set-theoretic operators. Then differentiable merge is defined by

$$A \diamond_\varepsilon B = \mathrm{pop}_1(U_\varepsilon(A, B)).$$

The resulting operator is fully differentiable provided U_ε and pop_τ are differentiable. This formulation provides a foundation for gradient-based optimization of geometric computations.

9.3 Collapse Flows and Gradient Propagation

The continuous collapse pop_τ induces a geometric flow defined by

$$\frac{d}{d\tau}A_\tau = V(A_\tau),$$

where V is a vector field representing the collapse direction in the chosen geometric representation. The flow interpretation permits differentiation through the collapse operator via the chain rule,

$$\frac{\partial A_1}{\partial A_0} = \int_0^1 DV(A_\tau)\, d\tau$$

provided the differential DV exists along the collapse trajectory. In practical settings, V may be implemented by gradient-based displacement or smoothing operations, permitting learned collapse dynamics.

9.4 Parametric Collapse Families

In learning systems, the collapse operator may be parameterized by learnable parameters θ, resulting in

$$\text{pop}_\theta : \mathcal{R}_d \to \mathcal{R}_d.$$

The objective is not solely to eliminate detail but to learn an abstraction function that preserves relevant structure. When pop_θ is differentiable in θ, we may optimize it using standard gradient methods. This promotes collapse from a fixed projection to a learnable abstraction, reflecting domain knowledge encoded through optimization rather than manual specification.

9.5 Continuous Merge–Collapse Semantics

The continuous merge–collapse operator is defined by

$$A \diamond_{\varepsilon,\theta} B = \text{pop}_\theta(U_\varepsilon(A, B)).$$

Varying ε and θ provides a continuum of computational behaviors ranging from strict geometric merge–collapse to soft blending and learnable abstraction. These continuous variants generalize Spherepop semantics to a differentiable computational calculus, enabling gradient propagation, backpropagation, and optimization while maintaining geometric intuition.

9.6 Analytic Equivalence and Smooth Approximations

Discrete merge–collapse semantics are recovered in the limit

$$\lim_{\varepsilon \to 0} \mathrm{pop}_\theta \circ U_\varepsilon = \mathrm{pop},$$

assuming θ is chosen to emulate canonicalization. In this manner, differentiable Spherepop constitutes a smooth approximation of the discrete calculus, and continuous semantics converge to classical Spherepop in the limit. The continuous formulation preserves expressive power while enabling analytic reasoning and optimization, establishing Spherepop as a geometric foundation for differentiable computation.

9.7 Discussion

Differentiable Spherepop bridges geometric computation with continuous optimization. Collapse becomes a smooth projection, merge becomes differentiable aggregation, and entire Spherepop programs may be differentiated end-to-end. This establishes a conceptual foundation for learning geometric computations, optimizing collapse rules, and integrating Spherepop into gradient-based frameworks while preserving its geometric character.

10 Expressiveness and Computability

We now examine the expressive capacity of Spherepop as a computational formalism. Earlier sections established its informal ability to encode logical and functional behavior. Here we construct precise encodings of classical models of computation, beginning with simple boolean circuits and proceeding toward encodings of lambda calculus and register machines. The purpose of this section is not only to assert computational power, but to articulate the manner in which geometric abstraction and merge–collapse dynamics implement the standard primitives of symbolic computation.

10.1 Boolean Algebra and Circuit Semantics

Boolean values in Spherepop are represented by the presence or absence of regions modulo collapse. Let T denote a canonical region representing truth, and let $\varnothing$ (or a collapsed absence) represent false. Boolean connectives may be defined through merge and controlled collapse, subject to structural regularity assumptions on collapse. For example,

$$A \vee B := \mathrm{pop}(A \cup B),$$

and suitable collapse conditions ensure that the result collapses to T exactly when one or both regions are nonempty. Logical conjunction may be defined by requiring that collapse be delayed until after mutual interaction, thereby establishing that

$$A \wedge B = \mathrm{pop}((A \cap B) \cup \mathrm{aux}),$$

where auxiliary constructions enforce collapse only when both operands contribute. Negation may be defined by complement regions (when such operations are permitted in the region space) or by controlled interaction with a reference region. These definitions yield a well-formed boolean algebra provided collapse respects canonicalization.

Boolean circuits are assembled by composing these primitives according to the geometry of merge–collapse interaction. A circuit becomes a spatial configuration of regions whose evaluation under Spherepop semantics corresponds to the logical outcome of the circuit.

10.2 Encoding Finite-State Computation

Finite-state automata may be encoded by associating each state with a distinct region, and transitions with controlled merge–collapse dynamics. Let $\{S_i\}$ denote canonical regions representing states. Given a transition $S_i \xrightarrow{a} S_j$, one constructs a merge pattern which, on interaction with input region A, collapses to S_j. Acceptance is determined by the canonicality of the terminal region. In this manner, Spherepop simulates any regular language by corresponding geometric control rules, demonstrating that the merge–collapse calculus subsumes finite-state computation.

10.3 Toward Turing Completeness

To establish Turing completeness, one must show that Spherepop can encode unbounded memory and simulate arbitrary rewrite sequences. Several routes exist. One approach is to encode lambda calculus, representing lambda abstraction by controlled collapse rules and application by merge. Another approach is to encode register machines or cellular automata by arranging regions in a spatial grid with collapse rules that enforce local update semantics.

We adopt the lambda-calculus approach, as it is the most direct generalization of the remark that collapse acts as abstraction and merge acts as application.

10.4 Lambda-Term Encoding

Let Λ be the set of lambda terms generated by

$$M ::= x \mid \lambda x.M \mid M\,N.$$

We define an encoding $\cdot_\Lambda : \Lambda \to \mathcal{R}_d$ mapping lambda terms to regions. Variables are represented by canonical primitive regions V_x, abstractions by controlled collapse families, and application by merge. Formally,

$$x_\Lambda = V_x, \qquad \lambda x.M_\Lambda = C_x(M_\Lambda), \qquad M\,N_\Lambda = M_\Lambda \diamond N_\Lambda,$$

where C_x collapses all contributions associated with x, yielding a canonical representation of the abstraction. The merge–collapse dynamics then implement β-reduction geometrically, removing the contribution of the bound variable and forming a collapsed region representing the result of substitution.

10.5 Simulation of β-Reduction

Let us sketch the simulation of β-reduction. Consider $(\lambda x.M)\,N$. Under the geometric encoding, merge produces a region in which V_x and N_Λ interact, and subsequent collapse erases the interior contribution of x in favor of the canonical representative of N. The result collapses to $M[N/x]_\Lambda$ up to geometric equivalence. Thus,

$$(\lambda x.M)\,N_\Lambda \rightsquigarrow M[N/x]_\Lambda,$$

which is precisely β-reduction. Iterated merge–collapse steps simulate repeated β-reductions. Termination requires care, but the encoding suffices to show that Spherepop can emulate lambda-calculus computation.

10.6 Computational Completeness

Because lambda calculus is Turing complete, and Spherepop simulates β-reduction up to geometric equivalence, it follows that Spherepop is computationally complete. Formally, any Turing machine computation may be encoded as a lambda term and evaluated in Spherepop under merge–collapse semantics. Thus, Spherepop is at least as expressive as the untyped lambda calculus.

Moreover, extensions to type theories, linear logic, and functional programming arise by restricting collapse and merge rules or by adding controlled interaction conditions. A detailed treatment of these systems is postponed to later sections.

10.7 Complexity Considerations

The computational complexity of Spherepop evaluation depends on the geometric representation of regions and the cost of merge and collapse. Under voxel representation, merge reduces to bitwise union and collapse to morphological smoothing, yielding polynomial-time operations. Under mesh representation, constructive solid geometry may increase complexity, but approximate methods reduce cost. The number of merge–collapse steps dominates runtime, and in general, the evaluation of arbitrary Spherepop programs can be computationally expensive. Precise complexity bounds require assumptions about collapse termination and geometric simplification routines, but the existence of exponential behavior follows from the encoding of lambda calculus.

10.8 Discussion

Spherepop is expressive enough to simulate standard computational systems, including boolean circuits, finite-state automata, and Turing machines. This expressiveness arises from the interpretation of merge as composition and collapse as abstraction, together with the capacity of regions to encode information in geometric form. The geometric style does not diminish computational power; instead, it provides a visual and spatial interpretation of classical computation.

11 Formal Semantics

We now present formal semantic foundations for Spherepop. The purpose of this section is twofold. First, we introduce a precise operational semantics for merge–collapse computation. Second, we develop a categorical semantics that abstracts geometric details and treats Spherepop computations as morphisms in a suitable category. The resulting formalism clarifies the meaning of computation in terms of algebraic structure, enabling comparison with existing models such as lambda calculus and Tensor Logic. The formal semantics are independent of particular geometric implementations and apply equally to discrete, continuous, and differentiable variants.

11.1 Operational Semantics

Let $\mathcal{R}$ denote a class of geometric regions satisfying appropriate regularity conditions (e.g. closed, bounded, connected). A Spherepop configuration is a finite multiset of regions

$$C = \{A_1, \ldots, A_n\}, \qquad A_i \in \mathcal{R}.$$

The state of a computation at time t is a configuration C_t. The operational semantics is defined by two transition rules:

$$C_t \longrightarrow C_{t+1}$$

whenever either a merge or a collapse transition applies. Merge transitions form the union of eligible regions, and collapse transitions replace regions by canonical representatives.

Formally, a merge transition takes $A_i, A_j \in C_t$ and produces

$$C_{t+1} = (C_t \setminus \{A_i, A_j\}) \cup \{\mathrm{pop}(A_i \cup A_j)\}.$$

A collapse transition takes A_i and replaces it with $\mathrm{pop}(A_i)$ whenever $\mathrm{pop}(A_i) \neq A_i$. Evaluation continues until no further merge or collapse rules apply, or diverges if an infinite sequence of transitions is possible.

11.2 Normal Forms

A configuration C is in *normal form* if $\mathrm{pop}(A) = A$ for every region $A \in C$ and no merge applies. Because collapse is idempotent, normal forms are fixed points of collapse. If merges are acyclic and collapse strictly reduces interior geometric complexity, evaluation terminates in finitely many steps. However, because Spherepop can encode lambda calculus (as proved earlier), termination is undecidable for arbitrary programs. Thus, evaluation is semantically well-defined but not guaranteed to halt.

11.3 Equivalence of Configurations

Different geometric configurations may represent the same computational state. To identify equivalent configurations, we introduce a semantic equivalence relation $\simeq$ on $\mathcal{R}$ generated by the axioms

$$A \simeq \mathrm{pop}(A), \qquad A \diamond B \simeq \mathrm{pop}(A \cup B).$$

Two configurations are equivalent if they differ only by these equivalences. In particular, all regions that collapse to the same canonical representative are considered semantically equal. The operational semantics induces equivalence classes of configurations that correspond to semantic values.

11.4 Denotational Semantics

We now define a denotational semantics that assigns to each Spherepop expression a semantic object in an abstract domain. Let $\mathcal{D}$ be a semantic domain of equivalence classes of regions under $\simeq$. For each primitive region A, let $[A] \in \mathcal{D}$ be its equivalence class. Define

$$A = [A], \qquad \mathrm{pop}(t) = t, \qquad t_1 \diamond t_2 = t_1 \sqcup t_2,$$

where $\sqcup$ denotes the induced binary operation on equivalence classes. Because collapse identifies values up to canonical representation, denotational semantics collapses syntactic distinctions that carry no semantic information. Thus, denotational semantics is a natural abstraction of operational computation.

11.5 Categorical Semantics

Spherepop may also be interpreted categorically. Let Sph be a category whose objects are semantic equivalence classes of regions and whose morphisms represent merge–collapse computations. Morphisms compose by merge, and identity morphisms correspond to regions that collapse to themselves. Formally, each object is an equivalence class $[A]$, and each morphism $f : [A] \to [B]$ corresponds to a region C such that

$$[B] = [A] \sqcup [C].$$

Composition is induced by merge,

$$g \circ f = f \diamond g,$$

and identity morphisms satisfy $A \diamond \mathrm{pop}(A) = A$. In this manner, Sph becomes a monoidal category under merge, with unit given by an empty or identity region. Collapse then induces a monoidal structure compatible with equivalence of objects.

11.6 Adjunctions and Quotient Structure

Collapse may be viewed as a quotient functor that identifies regions differing only by interior detail. Let Geo denote a geometric category whose objects are regions, and let Sph be the quotient category under $\simeq$. Then collapse induces a functor

$$Q : \mathsf{Geo} \to \mathsf{Sph}$$

that is surjective on objects and quotient on morphisms. The kernel of Q consists precisely of morphisms that differ only by internal geometric variation eliminated by collapse. This interpretation clarifies the semantic role of collapse and supports comparisons with Tensor Logic, where linear maps serve as morphisms and equivalence classes correspond to tensor identifications.

11.7 Soundness and Adequacy

Operational semantics is sound with respect to denotational semantics: if $t \rightsquigarrow t'$, then $t = t'$. Adequacy holds under the assumption that collapse preserves semantic equivalence, so that denotational semantics reflects operational behavior. In particular, normal forms correspond to canonical representatives in $\mathcal{D}$. The combination of soundness and adequacy establishes a precise connection between geometric computation and its semantic interpretation.

11.8 Discussion

Formal semantics clarifies the computational meaning of merge and collapse independently of specific geometric implementation. Operational semantics interprets Spherepop as iterative collapse of geometric configurations, denotational semantics interprets computation as evaluation into equivalence classes of regions, and categorical semantics interprets Spherepop as a monoidal category in which merge acts as composition. Each perspective provides a distinct mathematical account, demonstrating that Spherepop possesses the necessary structure to serve as a foundation for geometric computing.

12 Comparison with Tensor-Based Computational Models

We now turn to the relationship between Spherepop and tensor-based representations of computation. While the two frameworks appear conceptually distant—one geometric and collapse-based, the other algebraic and coordinate-based—they share deep structural correspondences. In particular, tensor operations may be understood as algebraic projections of geometric merge–collapse dynamics, and conversely Spherepop may be interpreted as a geometric enrichment of tensor computation. The goal of this section is not to reproduce existing tensor formalisms, but to articulate a precise correspondence between algebraic and geometric modes of computation.

12.1 Algebraic Representation of Merge

Merge combines geometric regions by union and collapse, producing a canonical representative. In tensor formalisms, combination of values is typically implemented by tensor addition, concatenation, multiplication, or contraction. The algebraic operation

$$u \otimes v$$

forms a structured composite of vectors u and v. By contrast, Spherepop forms a geometric composite and then collapses it. However, once collapse has eliminated internal detail, the resulting region may be represented by a point in an abstract embedding space. That abstract representation functions as a tensor if the embedding is linear and respects composition. Thus, tensor combination may be viewed as

$$E(A \diamond B) = E(A) \otimes E(B),$$

in the special case where E is an embedding from spheres to tensors that respects merge, at least up to equivalence. Not every embedding possesses this property, but sufficiently regular collapse yields a canonical representative enabling such an identification.

12.2 Collapse as Nonlinear Activation

In tensor-based neural computation, nonlinearity is supplied by activation functions. In Spherepop, collapse plays this role, projecting geometric unions into canonical forms. Algebraically, collapse induces a nonlinear map

$$E(\mathrm{pop}(A)) = \sigma(E(A)),$$

where σ is a nonlinear function (e.g. ReLU, sigmoid) determined by the collapse rule. Collapse thus serves as a nonlinearity consistent with tensor operations under suitable embeddings.

12.3 Tensor Contraction and Geometric Abstraction

Tensor contraction reduces a composite tensor to a lower-dimensional tensor by summation over indices. Conceptually, contraction eliminates internal structure while preserving global dependencies. Collapse performs a parallel operation: internal features of a region are eliminated while global structure is preserved. Hence, both contraction and collapse implement abstraction, but collapse is geometric rather than algebraic.

Formally, if T is a composite tensor, contraction may be written

$$\mathrm{contr}(T) = T \star K,$$

for some kernel K. Collapse may be written in analogous form

$$\mathrm{pop}(A) = A \star \Phi,$$

where Φ encodes canonicalization. Under this reading, tensor contraction and collapse implement analogous abstraction principles.

12.4 Linearization as Projection

Spherepop operations are geometric by construction. Tensor operations are linear algebraic by construction. The relation between the two frameworks may be interpreted through projection. Given a geometric region A, a tensor representation may be obtained by applying a linear embedding

$$E : \mathcal{R} \to \mathbb{R}^n,$$

mapping geometric objects to vectors. Under suitable regularity assumptions, merge–collapse dynamics project under E to algebraic operations on vectors. The algebraic structure is thus a quotient of the geometric structure. Conversely, geometric structure may be recovered only up to equivalence from the algebraic image.

12.5 Interpretation of Computation

Tensor representations interpret computation as algebraic manipulation of coordinate arrays. Spherepop interprets computation as geometric interaction of regions. Nevertheless, both frameworks encode abstraction, composition, and aggregation. Algebraic computation models information as coordinate vectors, while geometric computation models information as spatial configurations. Despite the difference in representation, both frameworks express function composition and abstraction, and therefore possess comparable expressive power.

12.6 Uniqueness of Collapse

A distinctive feature of Spherepop is that collapse is geometrically motivated and need not be linear in any algebraic representation. By contrast, tensor computation requires nonlinear activation to prevent reduction to linear transformations. Collapse naturally supplies a nonlinear structure that is compatible with geometric representation and admits a tensor analogue via embedding. This mechanism suggests that geometric abstraction may be a more primitive computational concept than algebraic nonlinearity.

12.7 Discussion

The comparison between geometric and algebraic computation reveals a complementary relationship: tensor operations may be regarded as algebraic projections of geometric merge–collapse dynamics, and Spherepop as a geometric enrichment of tensor computation. These correspondences help connect geometric models of computation with established algebraic techniques, without assuming that either framework is subordinate to the other. The contrast between spatial abstraction through collapse and algebraic abstraction through contraction illuminates the conceptual foundations underlying both models and suggests avenues for hybrid approaches.

13 Implementation Theory and Geometric Data Structures

This section develops a theoretical account of Spherepop implementation. Earlier presentations adopted an informal geometric viewpoint; we now investigate the algorithmic foundations of merge and collapse for practical deployment in computational geometry. Our objective is not to recommend a single implementation strategy, but to identify structural constraints that ensure correctness, computational well-definedness, and stable behavior across representations.

13.1 Representational Classes

A Spherepop implementation is determined by a class $\mathcal{G}$ of geometric objects, a merge operator $U : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ approximating set union, and a collapse operator $C : \mathcal{G} \to \mathcal{G}$ enforcing canonicalization. Admissible representation classes include spheres, ellipsoids, convex polytopes, axis-aligned bounding boxes, implicit surfaces, voxel representations, and meshes. Each representation yields distinct algorithmic properties and different asymptotic behavior of merge and collapse.

A representation class is said to be *computationally coherent* if U and C are computable at finite cost, and C is idempotent. Many geometric representations satisfy these conditions. Spherical and voxel classes are computationally coherent by construction. Mesh representations may require approximation schemes to enforce idempotent collapse.

13.2 Union and Canonicalization

Merge is defined abstractly by $A \diamond B = C(U(A, B))$. A naive implementation would require computing exact geometric unions; however, exact union for arbitrary shapes may be computationally expensive. Instead, many implementations approximate union by either (a) constructive solid geometry (CSG), (b) union of implicit level sets, or (c) bitwise union of voxel occupancy grids.

In computationally coherent settings, collapse C must satisfy $C \circ C = C$. Canonicalization often proceeds by contracting geometric representations along geometric flows, smoothing mesh features, or thresholding voxel occupancy. Theoretical correctness requires only that C identifies interior structure and stabilizes under iteration. Practical design balances computational tractability against representational fidelity.

13.3 Mesh-Based Realizations

Mesh representations treat regions as polyhedral objects embedded in $\mathbb{R}^3$, equipped with boundary and adjacency information. Union may be approximated via CSG, while collapse is implemented by mesh smoothing or decimation. Mesh-based collapse resembles geometric regularization methods in computational geometry, although mesh simplification must be performed carefully to ensure idempotence. Surface quality, complexity control, and topological preservation depend on collapse heuristics.

The primary difficulty arises from ensuring that collapse converges to a canonical representative independent of triangulation order. Approximate canonicalization is achievable using geodesic smoothing, Laplacian regularization, or convex-hull collapse. The convex hull is particularly well-suited because it guarantees idempotence and provides a computationally coherent collapse, albeit at the cost of discarding shape details. When interior structure is semantically irrelevant, convex-hull collapse yields a satisfactory canonical representation.

13.4 Voxel-Based Realizations

Voxel representations approximate regions by occupancy grids, enabling rapid computation of union via bitwise OR. Collapse reduces occupancy via thresholding or morphological operators, provided collapse operations stabilize after finite iterations. The computational cost scales with grid resolution, and memory resources can be substantial for high-dimensional grids. Nevertheless, voxel collapse remains conceptually straightforward and well-suited for parallelization, thus providing a practical foundation for large-scale Spherepop systems.

In high-performance settings, GPUs are well-suited for voxel-based computation. Merge and collapse map efficiently onto parallel bitwise operations or parallel morphological operators. In such cases, Spherepop evaluation may be implemented as a sequence of GPU kernels, highlighting its suitability for large-scale geometric computation.

13.5 Implicit Surface Realizations

Implicit surfaces represent regions as level sets of scalar functions $f(x) = 0$. This representation is particularly natural in differentiable Spherepop, as union operations correspond to differentiable blending of implicit functions and collapse corresponds to smoothing or reparameterization. Implicit collapse may be implemented via curvature-driven flows, yielding smooth level sets that converge to canonical representatives. Although implicit collapse is computationally intensive, its continuous formulation supports differentiability and analytic reasoning, and therefore constitutes a promising direction for research in differentiable geometric computation.

13.6 Data Structures for Efficient Evaluation

Efficient evaluation requires spatial indexing and caching of collapse operations. Common geometric data structures such as octrees, k-d trees, bounding-volume hierarchies (BVH), and spatial hash grids provide accelerated access to region adjacency, overlap detection, and merge scheduling. These structures significantly reduce pairwise intersection costs, and thus asymptotic complexity of merge scheduling. BVHs are well-established in rendering and physics simulation and readily adapted to Spherepop.

In practice, performance depends largely on the cost of collapse. Collapse may be amortized using caching or incremental update schemes, particularly if collapse eliminates most interior detail. For dynamic computations, hierarchical spatial decomposition reduces collapse cost by collapsing local neighborhoods first, then applying global canonicalization.

13.7 Complexity Analysis

The asymptotic complexity of merge operations depends on the representation class. For voxels with resolution n^3, voxel merge costs $O(n^3)$ in the worst case, although sparse voxel structures may reduce cost. Mesh-union complexity depends on polygon count, typically $O(k \log k)$ or higher, depending on the complexity of CSG operations. Implicit unions depend on the complexity of blending operations and level-set reinitialization.

Collapse complexity is dominated by canonicalization. In convex-hull collapse, time complexity scales with the cost of computing the convex hull, typically $O(k \log k)$ for k vertices; in voxel collapse, complexity depends on morphological filtering operations; in implicit collapse, complexity depends on PDE integration and reinitialization.

Overall evaluation runtime depends on the number of merge–collapse steps, analogous to the number of reduction steps in lambda calculus. Because Spherepop is computationally complete, worst-case evaluation complexity is unbounded; however, practical implementations may be designed to enforce syntactic restrictions or collapse limits that ensure termination or yield useful partial approximations.

13.8 Discussion

Implementation theory demonstrates that Spherepop accommodates multiple geometric representations, each with distinct computational properties. No single representation optimally balances expressiveness, efficiency, and theoretical regularity. Instead, Spherepop defines a semantic foundation within which different implementation strategies may evolve independently, united by collapse and merge as fundamental operations. This separation of semantics from implementation is crucial for connecting geometric computation with practical algorithmic realizations and for enabling a principled comparison of geometric computation with algebraic methods.

14 Differentiable Spherepop and Gradient-Based Optimization

The preceding sections have established discrete and continuous variants of merge and collapse. We now examine the analytic structure required to differentiate Spherepop computations and thus integrate them with gradient-based learning methodologies. In particular, we describe conditions under which collapse constitutes a differentiable projection, characterize merge as a differentiable blending operator, and show how gradient flow propagates through merge–collapse sequences. Our objective is not to reduce geometric computation to algebraic gradient flow, but to articulate a framework in which classical optimization techniques may be applied to geometric evolutions.

14.1 Differentiability of Collapse

Let $\mathcal{G}$ be a differentiable manifold of geometric regions parameterized by shape functions, meshes, implicit level sets, or voxel occupancy fields. A differentiable collapse operator is a map

$$C_\theta : \mathcal{G} \to \mathcal{G},$$

smooth in both its input and its parameter θ. Differentiability is understood in the Fréchet sense when $\mathcal{G}$ is modeled as a Banach space of implicit functions, or in the manifold sense when $\mathcal{G}$ represents a smooth surface or embedded submanifold. Under either representation, differentiability requires that small perturbations in the input region lead to small perturbations in the collapsed region.

The collapse operator C_θ must satisfy $C_\theta(C_\theta(A)) = C_\theta(A)$ (idempotence) and depends smoothly on θ, thereby enabling optimization of collapse rules by gradient descent. The functional form of C_θ may encode smoothing, radius contraction, curvature flow, or level-set reparameterization, each admitting differentiable approximations under appropriate discretizations.

14.2 Differentiable Merge

Merge is differentiable when the union operator is replaced by a differentiable blending operator,

$$U_\varepsilon : \mathcal{G} \times \mathcal{G} \to \mathcal{G},$$

smooth in both arguments and converging to geometric union in the limit as $\varepsilon \to 0$. The differentiable merge operator is then defined by

$$M_{\varepsilon,\theta}(A, B) = C_\theta\Big(U_\varepsilon(A, B)\Big),$$

smooth jointly in A, B, and (ε, θ). Differentiability allows gradient information to flow through merge operations just as in tensor-based neural layers, while retaining geometric semantics.

14.3 Gradient Flow Through Collapse

Let $\Phi : \mathcal{G} \to \mathbb{R}$ be a differentiable loss functional measuring the discrepancy between a Spherepop computation and a target. To optimize θ, one considers the composite map $\Phi \circ C_\theta$, whose derivative is

$$D_\theta(\Phi \circ C_\theta)(A) = D\Phi\Big(C_\theta(A)\Big) \circ D_\theta C_\theta(A).$$

Since C_θ is idempotent, its derivative must annihilate directions corresponding to collapsed information and preserve canonical directions. The resulting derivative is a projection onto the tangent space of canonical directions determined by collapse. This contrasts with classical neural networks, in which activation functions impose nonlinearity without geometric projection.

14.4 Gradient Propagation Through Merge–Collapse Sequences

For composite computations of the form

$$A_0 \mapsto A_1 = M_{\varepsilon,\theta}(A_0, B_0) \mapsto A_2 = M_{\varepsilon,\theta}(A_1, B_1) \mapsto \cdots,$$

gradient flow proceeds by repeated application of the chain rule,

$$D\Phi(A_k) = D\Phi(A_{k+1}) \circ DM_{\varepsilon,\theta}(A_k, B_k),$$

propagating sensitivity through both merge and collapse. Unlike tensor networks, gradient directions are constrained by geometric projection; collapse eliminates directions that correspond to collapsed interior states, ensuring that the gradient landscape is shaped by geometric abstraction. This yields a geometric analogue of backpropagation, in which learning corresponds to adapting collapse rules and merge interactions.

14.5 Analytic Collapse Flows

Continuous collapse flows may be defined by ordinary differential equations

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t} = V_\theta\Big(A(t)\Big), \qquad A(0) = A_0,$$

whose solution converges to a canonical fixed point

$$A_\infty = C_\theta(A_0).$$

Differentiability follows from standard existence and uniqueness theorems for ODEs, provided V_θ is smooth. Analytic collapse flows provide a principled foundation for differentiable canonicalization and allow gradient-based learning of collapse dynamics.

14.6 Optimization of Collapse Rules

Since C_θ reduces interior structure, optimization of θ amounts to learning a canonical abstraction function. In machine learning, abstraction functions are typically fixed, whereas in Spherepop they are learnable geometric projections. Learning a collapse function that preserves task-relevant structure offers a novel alternative to conventional feature extraction methods. Collapse becomes not merely a computational primitive, but a learned geometric abstraction mechanism whose semantics are domain-specific rather than predefined.

14.7 Discussion

Differentiable Spherepop establishes a continuous and analytic extension of geometric computation. Merge becomes differentiable aggregation, collapse becomes a differentiable abstraction operator, and Spherepop admits gradient-based optimization with geometric semantics preserved. This integration of geometric computing and differentiable programming suggests future research directions in geometric learning, differentiable geometry, and continuous computational semantics, and provides a conceptual alternative to tensor-oriented views of neural computation.

15 Complexity, Decidability, and Termination

The expressive power of Spherepop places it squarely among computationally complete formalisms. This section examines the corresponding theoretical implications, including computational complexity, decidability of termination, and asymptotic cost of evaluation. Our discussion emphasizes that the merge–collapse calculus, while conceptually geometric, inherits undecidability phenomena intrinsic to universal computation. At the same time, Spherepop admits meaningful complexity analyses under syntactic and geometric restrictions.

15.1 Undecidability of Termination

Let t be a Spherepop term and let $\rightsquigarrow$ denote one-step operational reduction. The termination problem asks whether evaluation

$$t \rightsquigarrow t_1 \rightsquigarrow t_2 \rightsquigarrow \cdots$$

halts after finitely many steps. Because Spherepop simulates untyped lambda calculus (Section 8), and termination in lambda calculus is known to be undecidable, it follows that termination in Spherepop is undecidable; that is, no algorithm can determine for all possible inputs whether a given Spherepop term terminates.

Formally, if T_Λ is the termination predicate for lambda calculus, and E is the encoding defined previously, then

$$T_\Lambda(M) \iff T_{\mathrm{Sph}}(E(M)),$$

where T_{Sph} denotes the termination predicate for Spherepop. Since T_Λ is undecidable, T_{Sph} is also undecidable. Thus, Spherepop inherits all classical undecidability properties of universal computation.

15.2 Worst-Case Complexity

Because Spherepop simulates lambda calculus, the worst-case cost of evaluating a Spherepop expression grows at least as fast as the worst-case cost of beta-reduction. In the absence of syntactic constraints, worst-case complexity cannot be bounded by any primitive recursive function. More precisely, we may encode lambda terms whose evaluation requires arbitrarily large numbers of merge–collapse steps, mimicking Turing machine behavior on hard instances.

At the same time, implementations of merge and collapse impose additional geometric costs. The complexity of a single merge depends on the geometric representation; voxel representations have cost proportional to grid size, mesh representations depend on polygonal complexity, and implicit-surface implementations depend on PDE integration. Thus, worst-case evaluation cost is bounded below by beta-reduction complexity and above by geometric costs imposed by merge–collapse operators.

15.3 Syntactic Restrictions and Guaranteed Termination

Certain syntactic fragments of Spherepop enjoy guaranteed termination. For example, if collapse is applied eagerly and each merge strictly reduces dimensional or volumetric measure, then evaluation must terminate in finitely many steps. Let $\mu(A)$ denote a geometric measure on region A. Assume collapse preserves μ and that merge satisfies

$$\mu(A \diamond B) < \mu(A) + \mu(B).$$

Inductively, repeated merge–collapse reduces the sum of measures and must terminate. Such geometric constraints enforce termination, but at the cost of reducing expressiveness. These fragments are analogous to strongly normalizing fragments of lambda calculus and linear logic.

15.4 Bounded Collapse Depth

Termination also follows when collapse depth is bounded. Define the collapse depth $d(t)$ as the number of nested collapse operations in t. If $d(t)$ is bounded by a constant independent of evaluation, then repeated application of collapse cannot iterate indefinitely. For instance, if C represents collapse and $C(C(A)) = C(A)$ by idempotence, then each collapse reduces or preserves depth, and cannot increase indefinitely. However, merge might reintroduce new collapsible regions; bounding collapse depth requires syntactic restrictions on the placement of merge operations.

15.5 Tractable Fragments

Practical implementations often adopt tractable fragments by enforcing one or more of the following:

- eager collapse after each merge,
- geometric bounds on collapse cost,
- bounded region volume in voxel representations,
- bounded polygon count in mesh collapse,
- restricted merge arity.

When such restrictions apply, evaluation proceeds in polynomial or quasi-polynomial time, depending on geometric complexity. In these implementations, Spherepop functions analogously to restricted lambda calculus or typed functional languages.

15.6 Parameterized Complexity

Complexity may be parameterized by the following quantities: the depth of merge chains, the number of primitive regions, the complexity of geometric representation (voxel resolution, mesh polygon count, implicit level-set resolution), and the analytic cost of collapse. Parameterized complexity theory allows incomplete evaluation under resource constraints, analogous to approximate reduction in functional programming or approximate inference in probabilistic programming.

For example, voxel Spherepop with resolution n^3 and bounded collapse radius admits evaluation in time $O(n^3)$ per collapse step. Under sparsity assumptions, complexity decreases significantly. Mesh implementations with bounded polygon count yield complexity proportional to surface complexity. Implicit implementations depend on PDE discretization and step size.

15.7 Discussion

Spherepop inherits the undecidability and worst-case complexity of classical computation, while admitting tractable behavior in restricted fragments. These results demonstrate that geometric models of computation enjoy expressive power equivalent to symbolic ones, and that geometric abstraction does not compromise theoretical strength. Future work should explore fragments with guaranteed termination, characterize expressive subsets, and develop resource-bounded semantics analogous to those in functional and logic programming.

16 Theoretical Properties, Open Problems, and Research Directions

The preceding development presents Spherepop as a computationally complete, semantically rigorous, and geometrically grounded model of computation. However, the model raises a number of theoretical questions whose answers would deepen its mathematical foundations and clarify its relationship to established formalisms. This section articulates these questions in formal terms and proposes directions for further research. The open problems fall into several categories, including semantic foundations, algebraic structure, computational properties, continuous variants, and applications to learning.

16.1 Hierarchy of Collapse Operators

A central theoretical question concerns the expressive hierarchy induced by collapse operators. Collapse plays multiple semantic roles: it enforces canonicalization, eliminates interior structure, and induces nonlinear computation. Different collapse operators yield different computational behaviors. One may ask whether there exists a hierarchy of collapse operators

$$C_0 \prec C_1 \prec C_2 \prec \cdots$$

ordered by expressive strength, analogous to hierarchies found in recursion theory or typed lambda calculi. It remains open whether stronger collapse operators yield strictly more expressive computational systems, or whether all reasonable collapse operators collapse to a common expressive ceiling.

16.2 Semantic Stability and Confluence

Confluence refers to the property that evaluation order should not affect the final semantic result. Since Spherepop computations may interleave merge and collapse in multiple orders, whether the resulting configuration is unique up to equivalence is a nontrivial question. Formally, given a term t and two evaluation sequences

$$t \rightsquigarrow^* t_1, \qquad t \rightsquigarrow^* t_2,$$

do we obtain $t_1 \simeq t_2$? If collapse is idempotent and merge satisfies suitable regularity conditions, then confluence holds under certain geometric assumptions. However, arbitrary collapse rules may violate confluence. Determining necessary and sufficient conditions for confluence constitutes a major open problem.

16.3 Geometric Invariants of Normal Forms

Normal forms are canonical configurations from which no further collapse or merge steps are possible. The geometry of normal forms remains poorly understood. Given a representation space $\mathcal{G}$ and collapse operator C, can we characterize the set

$$\mathcal{N} = \{A \in \mathcal{G} \mid C(A) = A\}$$

in geometric or topological terms? Are normal forms convex, smooth, contractible, or do they exhibit richer structure? Are normal forms unique representatives of equivalence classes, or can different canonical regions represent the same semantic value? These questions call for deeper analysis of collapse semantics and geometric invariants.

16.4 Resource-Bounded Semantics

General Spherepop computation admits undecidable termination. In practice, resource-bounded evaluation should impose limits on merge and collapse complexity. One may define a resource-sensitive semantics in which collapse is truncated after a fixed number of iterations or merge is restricted to bounded arity. The resulting semantics resemble bounded reduction in lambda calculus or bounded-resource models of algorithms. A resource-bounded theory of Spherepop would clarify practical implementations and provide connections to complexity theory.

16.5 Continuous Collapse and Differentiable Structure

Continuous and differentiable Spherepop raises questions concerning differentiable manifolds of regions. If $\mathcal{G}$ is a Banach manifold representing implicit surfaces or level sets, collapse becomes a projection onto a submanifold of canonical representatives. The geometry of this submanifold determines computational capacity. It remains unknown whether canonical submanifolds admit simple geometric characterizations or whether collapse generates intricate or fractal limiting surfaces. Analytical results may depend upon properties of differential equations governing collapse flows.

16.6 Monoidal and Cartesian Structure

Categorical semantics interpreted merge as a monoidal operator and collapse as a quotient functor. A precise identification of monoidal and Cartesian structure remains outstanding. If merge is associative up to isomorphism, Spherepop may admit monoidal closure; if collapse identifies internal detail, Spherepop may admit a Cartesian structure up to equivalence. Establishing formal categorical structure is crucial for comparing geometric computation with algebraic computation, and may reveal deep relationships between geometric collapse and abstract logical systems.

16.7 Comparison with Established Models

Although preliminary comparisons have been drawn, formal equivalences and inequivalences between Spherepop and established computational models remain to be investigated. Does Spherepop correspond to a known rewriting formalism? Can Spherepop be presented as a kind of interaction net, graph rewriting system, membrane computing model, or chemical reaction network? A comprehensive classification relative to known universality results would place geometric computation within the broader landscape of theoretical computer science.

16.8 Computational Geometry and Learning

Finally, differentiable Spherepop suggests applications in geometric learning, neural implicit representations, and differentiable rendering. Theoretical questions concerning approximation power, generalization, and optimization remain open. What kinds of functions can differentiable Spherepop approximate efficiently? Does Spherepop enjoy universal approximation properties analogous to neural networks? Can differentiable collapse learn canonical representations that outperform algebraic abstraction in traditional learning systems? These questions invite interdisciplinary research connecting computational geometry, differential equations, and machine learning.

16.9 Future Directions

Spherepop represents a geometric foundation for computation whose semantic structure is only partially understood. The integration of collapse and merge with classical notions of abstraction and composition provides a novel framework for reasoning about computation visually, geometrically, and formally. Future work should develop rigorous semantics, identify expressive boundaries, classify collapse operators, analyze normal forms, and establish confluence properties. The resulting theory may reveal geometric foundations underlying computation, and advance a unifying perspective in which abstraction, composition, and learning emerge from geometric interaction.

17 Conclusion

The purpose of this work has been to develop a rigorous theoretical framework for Spherepop as a geometric model of computation. While earlier presentations have emphasized pedagogical motivation or visual intuition, the present exposition has focused on formal semantics, computational expressiveness, geometric implementation, differentiable structure, and open theoretical questions. By treating merge as geometric union (or approximations thereof) followed by collapse as canonicalization, Spherepop defines computation as successive abstraction of geometric configurations. These abstractions preserve global semantic structure while eliminating interior detail, thereby inducing a natural notion of geometric computation that parallels—but does not depend on—algebraic representations.

We have argued that Spherepop is computationally complete and therefore inherits classical undecidability and complexity results. Nevertheless, Spherepop also admits meaningful geometric restrictions, enabling tractable, resource-bounded computation. The tension between expressive generality and practical implementation underscores the importance of geometric data structures, canonical collapse rules, differentiability conditions, and analytic flows for geometric canonicalization. These considerations place Spherepop at the intersection of computational geometry, lambda calculus, differentiated programming frameworks, and theoretical computer science.

Conceptually, geometric abstraction via collapse plays the same semantic role as contraction and activation in algebraic models while providing a continuous geometric interpretation. This duality motivates a deeper investigation of the relationships between geometric and algebraic computation, suggesting that traditional tensor-based representations may be specialized projections of more general geometric interactions. The theory developed here provides the foundations for such investigations, while identifying multiple open directions for future research, including categorical semantics, analytic characterization of canonical manifolds, complexity theory of collapse hierarchies, and differentiable learning in geometric spaces.

Spherepop therefore stands not merely as a computational system, but as an organizing principle unifying geometry, abstraction, and computation. Its geometric semantics invite a reconceptualization of computation itself, in which operations are not symbolic re-writings but geometric interactions governed by continuous flows and canonical reductions. Much work remains to establish the mathematical, algorithmic, and empirical implications of this perspective. The present exposition aims to lay the foundation for such work and to open a path toward a comprehensive geometric theory of computation.

A Notation and Conventions

Spherepop computations are expressed in terms of geometric regions, merge operations, and collapse operators. This appendix collects the principal notational conventions and semantic definitions used throughout the text.

A.1 Geometric Space

All geometric objects are elements of a representation space $\mathcal{G}$ modeled as a subset of a Euclidean space $\mathbb{R}^n$. Throughout the paper, $n = 3$ is assumed for concreteness, although lower or higher dimensional formulations are possible. Points in $\mathbb{R}^n$ are denoted by $x = (x_1, \ldots, x_n)$. A geometric region A is a subset of $\mathbb{R}^n$ equipped with additional structure depending on representation (e.g. boundary, mesh, implicit function, voxel occupancy).

A.2 Merge

Merge is denoted by the binary operator $\diamond$,

$$A \diamond B = C\Big(U(A, B)\Big),$$

where U is a geometric union operator and C is a collapse operator. Merge combines geometric objects by forming a union and then applying canonicalization. When context is clear, $A \diamond B$ may be written AB.

A.3 Collapse

Collapse is denoted by $C : \mathcal{G} \to \mathcal{G}$ and satisfies

$$C(C(A)) = C(A),$$

for all $A \in \mathcal{G}$. Collapse eliminates interior structure, smooths geometric detail, and identifies geometrically distinct regions according to a canonicalization rule. Different collapse operators yield different computational properties, and collapse may depend on additional parameters θ.

A.4 Operational Semantics

Evaluation steps are denoted by $\rightsquigarrow$. A Spherepop term is a finite composition of merge and collapse operations applied to primitive geometric regions. Evaluation proceeds by repeatedly applying merge–collapse until no further reduction is possible.

A.5 Equivalence

Two regions $A, B \in \mathcal{G}$ are semantically equivalent, written $A \simeq B$, if collapse yields the same canonical representative,

$$C(A) = C(B).$$

Equivalence classes define semantic values, while geometric representatives differ only up to collapse. Thus, Spherepop semantics identify regions modulo collapse.

A.6 Differentiability

Differentiable variants of merge and collapse are parameterized by (ε, θ) and denoted $M_{\varepsilon,\theta}$. Differentiability refers to smoothness in the Fréchet sense for implicit-surface representations, or smoothness on manifolds for mesh-based representations.

A.7 Measures

A geometric measure is denoted $\mu(A)$, representing volume, area, or another measure depending on dimension and representation. In certain fragments, merge strictly reduces or preserves measures, enforcing termination.

A.8 Evaluation

A computation is a sequence

$$t_0 \rightsquigarrow t_1 \rightsquigarrow t_2 \rightsquigarrow \cdots ,$$

terminating when a normal form is reached. If evaluation does not terminate, Spherepop exhibits the same non-termination phenomena as universal computation.

B Operational Semantics and Collapse Rules

This appendix provides a more formal account of merge–collapse semantics, including reduction rules, canonical representatives, and conditions for confluence.

B.1 Primitive Regions

Primitive regions are denoted by lowercase letters a, b, c and represent atomic spheres or geometric primitives. Without loss of generality, primitive regions are assumed to be closed and bounded.

B.2 Reduction by Merge

Operational reduction is defined by the rule

$$A \diamond B \rightsquigarrow C\Big(U(A, B)\Big).$$

If $U(A, B)$ contains interior geometric detail, collapse eliminates such detail by replacing $U(A, B)$ with a simpler canonical representative. Although $U(A, B)$ depends on representation, its operational semantics are explicitly defined by reduction to a canonical form.

B.3 Canonical Representatives

A region A is canonical if $C(A) = A$. Canonical representatives form a subspace $\mathcal{N} \subseteq \mathcal{G}$ of normal forms. Evaluation terminates precisely when a term is canonical. Canonical representatives depend on the choice of collapse operator and may vary across implementations.

B.4 Idempotence

Collapse rules must satisfy idempotence,

$$C(C(A)) = C(A),$$

ensuring that repeated collapse operations have no effect beyond the first. Idempotence implies that normal forms are fixed points of collapse and that collapse enforces unique representatives modulo equivalence.

B.5 Confluence (Conditional)

Evaluation is confluent if for any term t and evaluation sequences

$$t \rightsquigarrow^* t_1, \qquad t \rightsquigarrow^* t_2,$$

we have $t_1 \simeq t_2$. Confluence holds under suitable regularity assumptions on collapse, such as strict geometric contraction or convex canonicalization. In general, confluence is not guaranteed without such assumptions, and determining when collapse ensures confluence is an open problem.

C Implementation Representations and PDE Formulations

Collapse and merge may be implemented by discrete or continuous representations. This appendix outlines two principal implementation strategies—discrete voxel/surface representations and continuous implicit representations—and provides a general formulation of continuous collapse using PDEs.

C.1 Discrete Representation

Discrete implementations approximate geometric regions by voxels or meshes. Collapse may be implemented by thresholding or smoothing operators that eliminate internal detail. This approach is computationally tractable and parallelizable, but discrete collapse depends on resolution and introduces approximation error.

C.2 Implicit Representation

Implicit implementations represent geometric regions as zero level sets,

$$A = \{x \in \mathbb{R}^n \mid f(x) = 0\},$$

for some differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. Merge is approximated by blending implicit functions, e.g.

$$U(f, g) = \min(f, g),$$

while collapse corresponds to smoothing or reinitializing implicit functions via curvature flows.

C.3 Continuous Collapse via PDEs

Continuous collapse may be formulated as the solution of a partial differential equation,

$$\frac{\partial A}{\partial t} = -\kappa(A),$$

where κ is curvature or another functional that decreases interior detail. The solution converges to a canonical representative

$$A_\infty = C(A_0).$$

PDE-based collapse is differentiable, analytically tractable, and well-suited for continuous learning frameworks.

C.4 Hamiltonian and Lagrangian Formulations

Collapse may also be expressed in variational form. Given an energy functional

$$E(A) = \int_{\partial A} \mathcal{L}(A, \nabla A)\, ds,$$

collapse reduces $E(A)$ by gradient descent,

$$\frac{\partial A}{\partial t} = -\nabla E(A).$$

A corresponding Hamiltonian formulation treats collapse as evolution under a Hamiltonian vector field. These continuous formulations provide analytical tools for studying convergence and stability of collapse flows and suggest a geometric interpretation of computation as energy minimization.

References

[1] Cartan, Élie. *Geometry of Riemannian Spaces.* Princeton University Press.

[2] Needham, Tristan. *Visual Differential Geometry and Forms.* Princeton University Press, 2021.

[3] Grigor'yan, Alexander. *Heat Kernel and Analysis on Manifolds.* AMS, 2009.

[4] Osher, Stanley and Fedkiw, Ronald. *Level Set Methods and Dynamic Implicit Surfaces.* Springer, 2003.

[5] Bishop, Richard and Goldberg, Samuel. *Tensor Analysis on Manifolds.* Dover Publications.

[6] Sullivan, John M. "Curvatures of Smooth and Discrete Surfaces." *Discrete Differential Geometry*, Springer, 2008.