# Chapters 4–6: Finite Capacity-Based System

*Probabilistic Minds Consortium (voids.blog)*

2025

## Chapter 4: Emergence of Numbers and Arithmetic

### Introduction

In classical mathematics, numbers and arithmetic are often taken for granted as part of the continuum of the real line. However, in a strictly finite, capacity-based system, numbers emerge from the very building blocks that we have already set up—namely, the discrete probability increments tied to observer states. In this chapter, we explore how "numbers" and arithmetic operations can be constructed entirely from finite, rational increments. Instead of assuming a completed infinite set (like the real numbers), every number is built as a rational of the form $\frac{k}{N(a)}$ for some observer state $a$ and integer $k$.

This approach is very much in line with finitist and constructive mathematics, where every entity must be explicitly given by a finite process. For IT professionals, the benefits are immediate: the entire arithmetic framework is implemented with integer operations and fixed denominators, making it both exact and efficient on digital hardware. In addition, our method bears a strong resemblance to fixed-point arithmetic and common techniques used in digital signal processing, where numbers are represented in a discrete, bounded format.

The construction begins by identifying "intervals" between stable patterns and interpreting these intervals as finite segments. From these segments, we then form discrete rational numbers. Arithmetic operations—addition, subtraction, multiplication, and even division (when defined)—are carried out by "unifying denominators," much like the process taught in grade-school fraction arithmetic. The key difference is that every step is finite and based on capacity-dependent denominators, ensuring that the process never relies on any notion of an infinite or completed continuum.

Moreover, as the capacity $N(a)$ increases (see Chapter 3), the granularity of these fractions becomes finer. In practical terms, this means that while each observer state works with a finite set of rationals, a system that refines capacity can approximate classical rational arithmetic arbitrarily well without ever invoking actual infinity. The result is a framework in which the entire apparatus of arithmetic is "emergent" from the discrete structure—an approach that is both theoretically robust and well-suited to digital computation.

The following sections lay out the formal definitions and operations. We first explain how finite "intervals" naturally give rise to numbers and then describe how standard arithmetic operations can be performed in this discrete setting. Finally, we discuss why, despite the increased resolution available with capacity increases, no true infinite sets or irrational numbers ever appear in our system.

## 4.1 Constructing Increment-Based Numbers

### 4.1.1 Finite Intervals from Pattern Distances

- **Definition:**
  Suppose two stable patterns $x$ and $y$ are identified at a joint state $(a, e)$ with the property that the distance $\mathrm{Dist}(a, e, x, y)$ equals an integer $L$. We interpret this "segment" as a finite interval. The interval is defined as the set:

$$\left\{0,\, \frac{1}{L},\, \frac{2}{L},\, \ldots,\, 1\right\}.$$

  Each fraction $\frac{k}{L}$ represents a discrete step from $x$ to $y$.

- **Interpretation:**
  This construction allows us to view the segment as a scaled version of the probability increments available in the observer state. The process is entirely finite, with every fraction directly tied to a finite number $L$.

### 4.1.2 Emergence of Rational Numbers

- **Finite Rational Increments:**
  Since every probability in our system is given as $\frac{k}{N(a)}$ for some $a$ and $k$, these values themselves form a finite set of rational numbers. In this way, the set

$$\left\{0,\, \frac{1}{N(a)},\, \frac{2}{N(a)},\, \ldots,\, 1\right\}$$

  can be interpreted as the "number system" available to the observer at state $a$.

- **Comparison with Constructive Approaches:**
  In finitist mathematics, one constructs numbers step by step rather than assuming an infinite continuum. Our method aligns with this philosophy by ensuring that each number is an explicit, finite fraction. There is no notion of a "limit" in the classical sense—only a series of ever finer approximations as capacity increases.

## 4.2 Operations on Increment-Derived Numbers

### 4.2.1 Addition and Subtraction

- **Procedure:**
  To add or subtract two numbers $\frac{k}{L}$ and $\frac{m}{M}$, one first chooses an observer state $a'$ whose capacity is high enough that both denominators $L$ and $M$ can be refined to a common denominator (via the refinement process described in Chapter 3). Once both numbers are embedded into the finer resolution, addition is performed by simple integer addition:
  $$\frac{k'}{N(a')} + \frac{m'}{N(a')} = \frac{k' + m'}{N(a')}.$$
  Here, $k'$ and $m'$ are the appropriately scaled integers corresponding to $k$ and $m$.

### 4.2.2 Multiplication and Division

- **Multiplication:**
  The product of two increments $\frac{k}{N(a)}$ and $\frac{m}{N(a)}$ is defined as:
  $$\frac{k}{N(a)} \times \frac{m}{N(a)} = \frac{k \cdot m}{N(a)^2}.$$
  When needed, a refinement to a common capacity can be performed so that the result is again expressed in the standard form $\frac{k'}{N(a')}$.

- **Division:**
  Division is handled similarly to multiplication, with the caveat that division by zero is excluded. The process again requires refining to a common denominator and ensuring that the operation yields a rational number within the finite set.

### 4.2.3 Advantages for IT Implementation

- **Exact Arithmetic:**
  Since all operations are performed using integer arithmetic (or operations on finite rational numbers), there is no risk of rounding errors inherent in floating-point arithmetic. This is a significant advantage in computational settings where exactness is paramount.

- **Scalability via Capacity Refinement:**
  As higher precision is needed, one can simply transition to an observer state with a higher capacity $N(a')$. The previous arithmetic operations are preserved under the embedding, ensuring consistency and continuity in computations.

## 4.3 No Infinite Sets, No True Irrationals

### 4.3.1 Finite Data Guarantees

- **Key Principle:**
  Every numerical value, every arithmetic operation, and every intermediate result in our system is computed from finite sets. The set of all possible numbers in a given observer state is finite, which means there is no risk of unbounded computations.

### 4.3.2 Approximating Finer Values

- **Capacity Increases:**
  When an observer state transitions to a higher capacity, the finite set of numbers becomes denser. However, at no point do we invoke an infinite set; rather, we have a sequence of finite approximations that get arbitrarily close to any desired value.

- **No True Irrational Numbers:**
  All numbers in the system are rational by construction. While we can approximate the classical irrational numbers as closely as desired by refining capacity, they never appear as actual elements of our number system.

## Summary of Chapter 4

In this chapter, we have shown that the numbers and arithmetic operations used in our finite, capacity-based system emerge naturally from the discrete probability increments. By constructing finite intervals and performing arithmetic via common-denominator methods, we achieve a robust rational arithmetic that is both theoretically appealing and practically implementable. This approach mirrors key ideas in finitist mathematics while providing a solid foundation for computational applications.

# Chapter 5: Algebraic Structures from Finite Increments

## Introduction

Having developed a robust arithmetic framework in Chapter 4, we now turn our attention to the construction of algebraic structures—such as groups, rings, fields, and vector spaces—from the same finite, capacity-based increments. Classical algebra often presupposes infinite sets (for example, the field of rational numbers $\mathbb{Q}$ or the real numbers $\mathbb{R}$), but our approach is entirely finitist: every algebraic structure is constructed from a finite set of discrete fractions.

This chapter shows how our finite arithmetic, which emerges from the observer-dependent increments, can be extended to form structures with rich algebraic properties. We introduce the notion of "VOID Groups" as finite additive groups of increments, and we then build VOID Rings and VOID Fields by incorporating multiplicative operations. Finally, we discuss how these ideas extend naturally to vector spaces and matrices—core constructs in both pure mathematics and computer science.

Our approach is inspired by classical constructions such as $\mathbb{Z}/n\mathbb{Z}$ (the integers modulo $n$) and finite fields, but it is more flexible because it allows for refinement: as the observer capacity increases, the corresponding algebraic structures become "finer" and more detailed. For IT practitioners, the appeal is clear. Every operation is performed using finite data structures, ensuring both efficiency and exactness. Moreover, the ability to refine algebraic structures by increasing capacity is reminiscent of scalable digital systems, where upgrades in hardware or software naturally lead to higher precision and richer functionality.

In what follows, we first develop the additive structure of our increments (the VOID Group), then introduce multiplicative operations to form VOID Rings and VOID Fields. We also show how these finite fields serve as scalars for constructing finite-dimensional vector spaces and matrices. These algebraic constructs provide a powerful language for describing the behavior of our finite system, and they serve as a bridge between discrete mathematics and practical computational applications.

## 5.1 VOID Groups

### 5.1.1 The Finite Set of Increments

- **Definition:**
  For a given observer state $a$ with capacity $N(a)$, define the set of increments as:

$$\text{Increments}(a) = \left\{0,\ \frac{1}{N(a)},\ \frac{2}{N(a)},\ \ldots,\ 1\right\}.$$

To accommodate negative values, we extend this to:

$$\text{Increments}^{\pm}(a) = \left\{-1, -\frac{N(a)-1}{N(a)}, \ldots, 0, \ldots, \frac{N(a)-1}{N(a)}, 1\right\}.$$

- **Interpretation:**
  This finite set forms the basis for all subsequent algebraic structures. Each element is a discrete rational number that can be manipulated via arithmetic operations.

#### 5.1.2 Group Operations

- **Addition and Inverses:**
  Define the group operation as standard addition:

$$\frac{k}{N(a)} + \frac{m}{N(a)} = \frac{k+m}{N(a)}.$$

  The additive identity is $\frac{0}{N(a)}$, and each element $\frac{k}{N(a)}$ has an inverse $\frac{-k}{N(a)}$.

- **Associativity and Closure:**
  Since the set $\text{Increments}^{\pm}(a)$ is finite and closed under the above operations, it forms a finite abelian (commutative) group—what we call a "VOID Group."

### 5.2 VOID Rings and Fields

#### 5.2.1 Introducing Multiplication

- **Definition:**
  For two increments $\frac{k}{N(a)}$ and $\frac{m}{N(a)}$, define their product as:

$$\frac{k}{N(a)} \times \frac{m}{N(a)} = \frac{k \cdot m}{N(a)^2}.$$

  With an appropriate refinement (i.e., by considering a higher-capacity state $a'$ where $N(a')$ is a multiple of $N(a)$), the product can be expressed in the standard form.

#### 5.2.2 Ring and Field Properties

- **Ring Structure:**
  Together with the addition defined in Section 5.1, these operations yield a ring structure on the finite set of increments. The distributive laws hold, and the multiplicative identity is $\frac{N(a)}{N(a)}$.

- **Field Structure:**
  Under further refinement, for nonzero elements, a multiplicative inverse exists (at least within a suitably refined capacity). This gives rise to what we term a "VOID Field." The structure resembles that of a finite subfield of the rationals, constructed entirely from finite data.

## 5.3 Vector Spaces and Matrices

### 5.3.1 Finite Vector Spaces

- **Definition:**
  Let the VOID Field serve as the set of scalars. A finite vector space is then defined as an $n$-tuple $(v_1, v_2, \ldots, v_n)$ where each $v_i \in \mathrm{Increments}(a)$. Vector addition and scalar multiplication are defined componentwise.

- **Properties:**
  The resulting structure behaves like a classical finite-dimensional vector space but is entirely built from discrete, capacity-dependent increments.

### 5.3.2 Matrices and Linear Maps

- **Matrix Representation:**
  A matrix is defined as an array (or table) whose entries are elements of $\mathrm{Increments}(a)$. Standard operations (addition and multiplication) are performed using the finite arithmetic developed earlier.

- **Practical Implications:**
  These structures are directly implementable in software and hardware, and they maintain exactness because all operations are performed on finite sets. Refinement to higher capacities minimizes rounding errors and improves resolution, much like upgrading numerical precision in computational systems.

## Summary of Chapter 5

In Chapter 5, we have shown how algebraic structures—VOID Groups, VOID Rings, VOID Fields, vector spaces, and matrices—can be constructed from the same finite increments that underlie our arithmetic. This construction is entirely consistent with finitist principles: every structure is finite, every operation is exact, and the possibility of refinement guarantees that the system can achieve arbitrarily high precision without invoking actual infinities. These algebraic structures not only mirror classical constructs but also

have direct applications in computational settings, providing a bridge between abstract mathematics and practical algorithms.

# Chapter 6: Polygons, Polyhedra, and Topological Constructs

## Introduction

Having established a comprehensive arithmetic and algebraic framework based on finite increments, we now turn to the realm of discrete geometry. Chapter 6 is devoted to demonstrating how concepts such as polygons, polyhedra, and more general topological constructs emerge naturally from the finite and combinatorial nature of our system. Traditional geometry typically relies on the continuity of the Euclidean plane, but in our finite setting, geometry is defined in terms of adjacency, discrete distances, and finite combinatorial structures.

The starting point is the observation that stable patterns (introduced in Chapter 2) can be viewed as "points" in a discrete configuration space. By utilizing the finite metric defined on these points, we can construct geometric notions such as lines, loops, and higher-dimensional complexes. A "line" is simply a sequence of adjacent patterns where each "step" corresponds to the smallest possible change (i.e., a unit increment in our discrete metric). Similarly, a polygonal loop is a closed cycle of patterns that partitions the underlying graph into distinct regions—an idea that echoes the classical Jordan curve theorem but in a fully combinatorial manner.

This discrete approach to geometry has many parallels in digital topology and combinatorial geometry, where the focus is on explicitly definable structures rather than on abstract continuous spaces. For IT practitioners, the appeal is obvious: every geometric object is represented by a finite set of data, making the system well-suited for computer graphics, network analysis, and even robotics. Moreover, the method of refining geometric structures by increasing capacity (as discussed in Chapter 3) provides a systematic way to "zoom in" on details, much like increasing the resolution of an image.

In the following sections, we will define polygonal loops, describe how they induce an inside/outside partition of the discrete space, and extend these ideas to higher-dimensional constructs such as polyhedra. Our presentation is both rigorous and accessible, aimed at illustrating that even rich geometric and topological concepts can be built from purely finite, combinatorial ingredients.

## 6.1 Polygonal Loops

### 6.1.1 Defining a Polygonal Loop

- **Definition:**
  Consider a sequence of stable patterns $(x_1, x_2, \ldots, x_n)$ at a joint state $(a, e)$. We say that these patterns form a polygonal loop if the distance between consecutive

patterns is exactly one unit:

$$\mathrm{Dist}(a, e, x_i, x_{i+1}) = 1$$

and the loop closes with

$$\mathrm{Dist}(a, e, x_n, x_1) = 1.$$

- **Interpretation:**
  This cycle of patterns defines a closed path in the finite metric space. In digital or network terms, it is analogous to a cycle in a graph where every edge represents the minimal change between adjacent nodes.

#### 6.1.2 Inside/Outside Partition

- **Concept:**
  Once a polygonal loop is defined, it naturally partitions the graph into "inside" and "outside" regions. This is achieved by removing the edges of the loop and then using standard reachability algorithms (e.g., flood-fill) to determine which nodes are enclosed by the loop.

- **Analogy with Classical Topology:**
  Although our space is entirely discrete, the resulting partitioning is reminiscent of the Jordan curve theorem, which asserts that a simple closed curve in the plane divides it into an interior and exterior region. Here, the partition is achieved without any recourse to continuity or infinite sets.

## 6.2 Higher-Dimensional Constructs

#### 6.2.1 Extending to Polyhedra and Complexes

- **From Loops to Faces:**
  A face in our discrete setting is defined as a polygonal loop that bounds a two-dimensional region in the graph. By combining multiple faces, one can construct polyhedra (three-dimensional analogues) or more general complexes.

- **Vertices, Edges, and Faces:**

  - **Vertices:** These are the stable patterns (points) identified in our system.
  - **Edges:** Minimal-increment adjacencies between patterns act as edges.
  - **Faces:** Cycles (polygonal loops) form faces that can be combined to form polyhedral structures.

- **Combinatorial Topology:**
  The entire construction is combinatorial in nature. There is no need for continuous coordinates—only the finite set of patterns and the discrete distances between them. This approach echoes methods used in computational geometry and digital image processing.

#### 6.2.2 Applications in IT

- **Graph-Based Modeling:**
  The concepts of polygonal loops and polyhedra are directly applicable in areas such as computer graphics, where objects are modeled using meshes, and in network topology, where regions of a network may be defined by cycles.

- **Efficient Computation:**
  Since every object is finite and every operation is local (based on discrete steps), the resulting topological constructs can be stored, manipulated, and visualized using standard data structures and algorithms. Refinement via capacity increases enables higher resolution models without altering the fundamental discrete nature of the system.

## 6.3 Summary of Chapter 6

In Chapter 6, we have extended our finite framework to the realm of geometry and topology. By defining polygonal loops and higher-dimensional complexes in a discrete setting, we have shown that rich geometric structures can be built entirely from finite, combinatorial data. These constructs are not only theoretically sound but also directly applicable in computational environments. They provide an optimistic, robust alternative to classical continuous geometry, demonstrating that even the most sophisticated topological ideas can emerge from the finite, capacity-based approach.

## Final Remarks on Chapters 4–6

Across Chapters 4 to 6, we have witnessed a remarkable journey: starting from discrete probability increments, we built up an entire arithmetic framework, developed rich algebraic structures, and finally emerged with sophisticated geometric and topological constructs—all while remaining strictly within the realm of finite mathematics. This capacity-based approach, grounded in finitist principles, is not only mathematically rigorous but also ideally suited for modern IT applications. Every operation is computable,

every structure is stored in finite data, and every refinement step enhances resolution without invoking any notion of infinity.

The synthesis of arithmetic, algebra, and geometry in this discrete framework offers a fresh and optimistic perspective on how traditional mathematical concepts can be reinterpreted and implemented in a digital world. Whether you are a mathematician intrigued by finitist methods or an IT professional seeking robust, exact, and scalable models, these chapters provide a compelling blueprint for further exploration and application.