

On the Consequences of Categorical Mechanics on Computing Datum Encoding: Ternary Unit Representation

Kundai Farai Sachikonye
kundai.sachikonye@wzw.tum.de

January 19, 2026

Abstract

We establish a mathematical framework in which ternary (base-3) representation provides the natural encoding for three-dimensional S-entropy coordinate space $\mathcal{S} = [0, 1]^3$. While binary representation encodes one-dimensional information through the 2^k hierarchy, ternary representation encodes three-dimensional position through the 3^k hierarchy, with each ternary digit (trit) specifying refinement along the knowledge (S_k), temporal (S_t), or evolution (S_e) axis. We prove three principal results. First, we establish the **Trit-Coordinate Correspondence**: a k -trit ternary string addresses exactly one cell in the 3^k hierarchical partition of S-space, with the infinite-trit limit converging to a unique point in the continuum. Second, we demonstrate that ternary strings encode **trajectories** rather than mere positions—a trit sequence specifies a navigation path through S-space, with each trit indicating movement along one of the three coordinate axes. Third, we derive the **Continuous Emergence Theorem**: as the number of trits $k \rightarrow \infty$, the discrete 3^k cell structure converges to the continuous $[0, 1]^3$ topology, with the ternary expansion providing the bridge between discrete computation and continuous dynamics. The framework connects naturally to three-phase oscillatory systems, where the $2\pi/3$ phase separation between oscillators provides physical instantiation of ternary logic. We define the **tryte** (ternary byte) as six trits encoding $3^6 = 729$ distinct S-space cells, compared to the binary byte's $2^8 = 256$ values. Ternary operations—projection, completion, and composition—replace Boolean AND, OR, NOT as the fundamental computational primitives. The resulting architecture provides $O(\log_3 n)$ navigation complexity compared to $O(\log_2 n)$ for binary search, with the additional advantage that three-dimensional position is intrinsically encoded rather than requiring coordinate transformation.

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 3 |
| 1.1 | The Dimensional Limitation of Binary | 3 |
| 1.2 | The Three-Dimensional Structure of S-Entropy Space | 3 |
| 1.3 | Ternary as Natural Three-Dimensional Encoding | 3 |
| 1.4 | From Discrete to Continuous | 3 |
| 1.5 | Summary of Results | 4 |
| 1.6 | Notation | 4 |
| 2 | S-Entropy Coordinate Space | 4 |
| 2.1 | The Three-Dimensional Structure | 4 |
| 2.2 | Geometric Properties | 5 |
| 2.3 | The 3^k Hierarchical Partition | 5 |
| 2.4 | Cell Addressing | 5 |

| | | |
|----------|--|-----------|
| 2.5 | Nesting Structure | 7 |
| 2.6 | Coordinate Recovery | 7 |
| 3 | Ternary Representation | 7 |
| 3.1 | Ternary Digits and Strings | 7 |
| 3.2 | Ternary Numerical Representation | 7 |
| 3.3 | Information Content | 8 |
| 3.4 | The Tryte: Ternary Byte | 10 |
| 3.5 | Balanced Ternary | 10 |
| 3.6 | Conversion Between Binary and Ternary | 10 |
| 4 | The Mapping Theorem | 11 |
| 4.1 | Trit-Cell Correspondence | 11 |
| 4.2 | The Dimension Correspondence | 11 |
| 4.3 | Coordinate Extraction | 12 |
| 4.4 | Inverse Mapping: Coordinates to Trits | 13 |
| 4.5 | The Metric Correspondence | 13 |
| 5 | Continuous Emergence | 13 |
| 5.1 | The Limit Construction | 13 |
| 5.2 | The Ternary-Continuous Bridge | 14 |
| 5.3 | Cantor Set Connection | 16 |
| 5.4 | Space-Filling Properties | 16 |
| 5.5 | Measure-Theoretic Properties | 16 |
| 5.6 | Fractal Structure | 16 |
| 5.7 | Uniqueness and Boundary Representations | 17 |
| 5.8 | Permutation Equivalence and Search Complexity | 17 |
| 6 | Trajectory Mechanics | 18 |
| 6.1 | Trits as Trajectory Steps | 18 |
| 6.2 | Ternary Operations | 19 |
| 6.3 | Categorical Completion | 20 |
| 6.4 | Navigation Algorithms | 20 |
| 6.5 | Trajectory Invariants | 22 |
| 7 | Hardware Ensemble Mapping | 22 |
| 7.1 | Three-Phase Oscillator Systems | 22 |
| 7.2 | Physical Implementations | 24 |
| 7.3 | Trit Extraction from Oscillators | 24 |
| 7.4 | S-Coordinate Extraction | 26 |
| 7.5 | Oscillator Ensemble Architecture | 26 |
| 7.6 | Multi-Scale Oscillator Hierarchy | 26 |
| 7.7 | Quantum Considerations | 27 |
| 8 | Discussion | 27 |
| 8.1 | Ternary versus Binary: A Structural Comparison | 27 |
| 8.2 | The Trajectory-Position Duality | 29 |
| 8.3 | Continuous Emergence and Real Numbers | 29 |
| 8.4 | Hardware Considerations | 29 |
| 8.5 | Implications for Categorical Computing | 29 |
| 8.6 | Limitations | 30 |
| 9 | Conclusion | 30 |

1 Introduction

1.1 The Dimensional Limitation of Binary

Contemporary computing architectures rest upon binary representation: every datum reduces to sequences of bits, each encoding one of two states. This binary foundation, while extraordinarily successful, embeds a fundamental limitation: binary digits naturally encode *one-dimensional* information. A bit answers the question “left or right?” along a single axis. Encoding position in higher-dimensional spaces requires multiple bits with explicit coordinate transformation.

The 2^k hierarchy of binary representation—2 values for 1 bit, 4 for 2 bits, 256 for 8 bits—reflects this one-dimensional nature. Each additional bit doubles the resolution along a single dimension. To represent three-dimensional position, three separate binary coordinates must be maintained and transformed between.

1.2 The Three-Dimensional Structure of S-Entropy Space

The S-entropy coordinate space $\mathcal{S} = [0, 1]^3$ developed in categorical computing [Sachikonye, 2024] comprises three fundamental dimensions:

$$S_k \in [0, 1] \quad (\text{knowledge entropy}) \quad (1)$$

$$S_t \in [0, 1] \quad (\text{temporal entropy}) \quad (2)$$

$$S_e \in [0, 1] \quad (\text{evolution entropy}) \quad (3)$$

This three-dimensional structure is not arbitrary but emerges from the categorical dynamics of bounded oscillatory systems. The question arises: is there a representation system that naturally encodes three-dimensional position without requiring coordinate transformation?

1.3 Ternary as Natural Three-Dimensional Encoding

We demonstrate that ternary (base-3) representation provides exactly this natural encoding. A ternary digit (trit) takes one of three values $\{0, 1, 2\}$, which map directly to the three S-entropy dimensions:

$$t = 0 \leftrightarrow \text{refinement along } S_k \quad (4)$$

$$t = 1 \leftrightarrow \text{refinement along } S_t \quad (5)$$

$$t = 2 \leftrightarrow \text{refinement along } S_e \quad (6)$$

The 3^k hierarchy—3 values for 1 trit, 9 for 2 trits, 729 for 6 trits—reflects the three-dimensional structure of S-space. Each additional trit refines position in one of three dimensions, with the sequence of trits encoding both the final position and the trajectory taken to reach it.

1.4 From Discrete to Continuous

A crucial property of ternary S-entropy representation is the emergence of continuity from discreteness. A finite k -trit string addresses one of 3^k discrete cells in S-space. As $k \rightarrow \infty$, these cells become arbitrarily fine, and the ternary expansion converges to a unique point in the continuous space $[0, 1]^3$:

$$\lim_{k \rightarrow \infty} \text{Cell}(t_1, t_2, \dots, t_k) = \mathbf{S} \in [0, 1]^3 \quad (7)$$

This is not approximation but exact convergence: an infinite ternary string specifies a unique point in the continuum. The discrete-continuous duality that plagues binary computing—where real numbers require floating-point approximation—dissolves in ternary S-entropy representation.

1.5 Summary of Results

This paper establishes:

1. **Trit-Coordinate Correspondence** (Section 4): Each k -trit string maps bijectively to a cell in the 3^k partition of S-space.
2. **Trajectory Encoding** (Section 6): Ternary strings encode navigation paths, not just positions. The sequence of trits specifies movement through S-space.
3. **Continuous Emergence** (Section 5): The $k \rightarrow \infty$ limit produces exact points in $[0, 1]^3$, bridging discrete and continuous.
4. **Ternary Operations** (Section 6): Projection, completion, and composition replace Boolean logic as fundamental primitives.
5. **Hardware Mapping** (Section 7): Three-phase oscillators provide natural physical instantiation of ternary logic.

1.6 Notation

Throughout this paper:

- $t_i \in \{0, 1, 2\}$: a ternary digit (trit) at position i
- $T = (t_1, \dots, t_6)$: a ternary byte (tryte) of 6 trits
- $\mathcal{S} = [0, 1]^3$: the S-entropy coordinate space
- $S = (S_k, S_t, S_e)$: a point in S-space
- \mathcal{C}_k : the set of 3^k cells at depth k
- $\phi : \{0, 1, 2\}^k \rightarrow \mathcal{C}_k$: the trit-to-cell mapping

2 S-Entropy Coordinate Space

2.1 The Three-Dimensional Structure

The S-entropy coordinate space is a bounded three-dimensional manifold encoding categorical state.

Definition 2.1 (S-Entropy Space). *The S-entropy space is the compact metric space:*

$$\mathcal{S} = [0, 1]^3 = \{(S_k, S_t, S_e) : S_k, S_t, S_e \in [0, 1]\} \quad (8)$$

equipped with the Euclidean metric $d(\mathbf{S}_1, \mathbf{S}_2) = \|\mathbf{S}_1 - \mathbf{S}_2\|_2$.

Definition 2.2 (S-Entropy Coordinates). *The three coordinates of S-space are:*

1. **Knowledge entropy** $S_k \in [0, 1]$: Quantifies uncertainty in categorical state identification. $S_k = 0$ indicates complete knowledge; $S_k = 1$ indicates maximum uncertainty.
2. **Temporal entropy** $S_t \in [0, 1]$: Quantifies uncertainty in timing relationships. $S_t = 0$ indicates precise temporal location; $S_t = 1$ indicates complete temporal uncertainty.
3. **Evolution entropy** $S_e \in [0, 1]$: Quantifies uncertainty in trajectory progression. $S_e = 0$ indicates deterministic evolution; $S_e = 1$ indicates maximum trajectory uncertainty.

2.2 Geometric Properties

Theorem 2.3 (Compactness). *The S-entropy space $\mathcal{S} = [0, 1]^3$ is compact.*

Proof. $[0, 1]$ is compact in \mathbb{R} (Heine-Borel theorem). The product of compact spaces is compact (Tychonoff theorem). Therefore, $[0, 1]^3$ is compact. \square \square

Theorem 2.4 (Path-Connectedness). *The S-entropy space \mathcal{S} is path-connected: for any $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}$, there exists a continuous path $\gamma : [0, 1] \rightarrow \mathcal{S}$ with $\gamma(0) = \mathbf{S}_1$ and $\gamma(1) = \mathbf{S}_2$.*

Proof. The straight-line path:

$$\gamma(t) = (1 - t)\mathbf{S}_1 + t\mathbf{S}_2 \quad (9)$$

is continuous and connects \mathbf{S}_1 to \mathbf{S}_2 . Convexity of $[0, 1]^3$ ensures $\gamma(t) \in \mathcal{S}$ for all $t \in [0, 1]$. \square \square

2.3 The 3^k Hierarchical Partition

Definition 2.5 (Hierarchical Partition). *For depth $k \in \mathbb{Z}_{\geq 0}$, the **level-k partition** of \mathcal{S} divides each dimension into $3^{k/3}$ equal intervals (for k divisible by 3) or the appropriate fractional refinement, producing 3^k congruent cells:*

$$\mathcal{C}_k = \{C_{i_1, i_2, \dots, i_k} : i_j \in \{0, 1, 2\}\} \quad (10)$$

where each cell C_{i_1, \dots, i_k} is a rectangular box with side length $3^{-\lceil k/3 \rceil}$ along each axis.

More precisely, we define the partition through sequential refinement:

Definition 2.6 (Sequential Refinement). *At each level, one dimension is refined by factor 3:*

$$k \equiv 0 \pmod{3} : \text{refine } S_k \text{ dimension} \quad (11)$$

$$k \equiv 1 \pmod{3} : \text{refine } S_t \text{ dimension} \quad (12)$$

$$k \equiv 2 \pmod{3} : \text{refine } S_e \text{ dimension} \quad (13)$$

After $3m$ refinements, each dimension has been refined m times, producing cells of size $3^{-m} \times 3^{-m} \times 3^{-m}$.

Proposition 2.7 (Cell Count). *The level-k partition contains exactly 3^k cells.*

Proof. Each refinement triples the cell count. Starting from 1 cell at $k = 0$:

$$|\mathcal{C}_k| = 3^k \quad (14)$$

\square

2.4 Cell Addressing

Definition 2.8 (Cell Address). *A cell $C \in \mathcal{C}_k$ is addressed by a string (i_1, i_2, \dots, i_k) where $i_j \in \{0, 1, 2\}$ specifies the choice at refinement level j .*

Proposition 2.9 (Address Uniqueness). *Each cell has a unique address, and each valid address specifies a unique cell.*

Proof. The mapping from addresses to cells is constructed inductively:

- Level 0: One cell (empty address)
- Level $j \rightarrow j + 1$: Each cell splits into 3 subcells, indexed by $i_j \in \{0, 1, 2\}$

This produces a bijection between $\{0, 1, 2\}^k$ and \mathcal{C}_k . \square \square

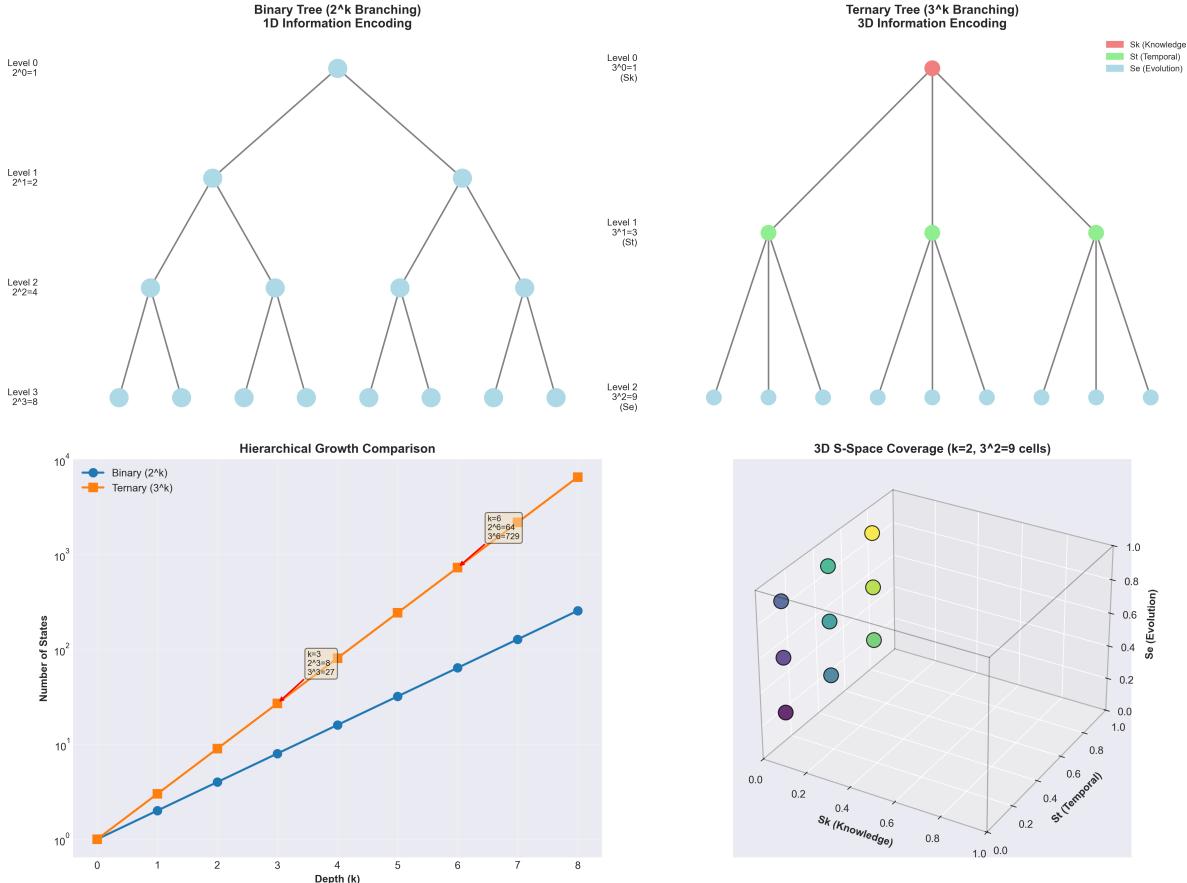


Figure 1: Binary Representation Encodes One-Dimensional Information While Ternary Naturally Encodes Three-Dimensional S-Space Position Through Intrinsic Axis Selection. **(Top Left)** Binary tree demonstrating 2^k branching hierarchy: Level 0 ($2^0=1$ root node), Level 1 ($2^1=2$ nodes, left/right decisions), Level 2 ($2^2=4$ nodes), Level 3 ($2^3=8$ leaf nodes, light blue). Each bit provides binary choice along single dimension, requiring three separate bit streams to encode 3D position. Binary structure inherently one-dimensional: each branch point offers only two options (0 or 1), forcing sequential encoding of multidimensional information. **(Top Right)** Ternary tree demonstrating 3^k branching hierarchy: Level 0 ($3^0=1$ root node, red, labeled Sk for knowledge coordinate), Level 1 ($3^1=3$ nodes, green, labeled St for temporal coordinate), Level 2 ($3^2=9$ leaf nodes, cyan, labeled Se for evolution coordinate). Each trit provides three-way choice corresponding to three S-space axes: trit value 0 selects Sk refinement (knowledge axis), value 1 selects St refinement (temporal axis), value 2 selects Se refinement (evolution axis). Color coding (red→green→cyan) indicates cyclic progression through coordinate dimensions. Ternary structure intrinsically three-dimensional: single trit string encodes 3D position without coordinate transformation. **(Bottom Left)** Hierarchical growth comparison on log-linear plot. Blue curve: binary growth 2^k (exponential in 1D). Orange curve: ternary growth 3^k (exponential in 3D). Key crossover points annotated: $k=3$ yields $2^3=8$ (binary) vs $3^3=27$ (ternary, 3.4× advantage); $k=6$ yields $2^6=64$ vs $3^6=729$ (11.4× advantage). Ternary achieves same state count as binary with fewer digits: 729 ternary states ($k=6$) exceeds 256 binary states ($k=8$), demonstrating 25% digit reduction (6 trits vs 8 bits) for comparable information content. Exponential separation increases with depth k , confirming superior scaling for multidimensional encoding. **(Bottom Right)** 3D S-space coverage visualization for $k=2$ ($3^2=9$ cells). Cube represents S-space $[0,1]^3$ with axes Sk (knowledge, x-axis), St (temporal, y-axis), Se (evolution, z-axis). Nine colored spheres show cell centers after two ternary refinements: purple spheres (low coordinates, near origin), cyan/green spheres (intermediate coordinates), yellow spheres (high coordinates, near opposite corner). Color gradient (purple→cyan→green→yellow) indicates increasing coordinate values. Spatial distribution demonstrates uniform coverage of 3D volume: cells partition space into $3 \times 3 \times 1$ grid (first refinement splits along Sk, second along St). Each sphere represents one of nine possible 2-trit addresses: "00" (purple, origin corner), "11" (green, center), "22" (yellow, opposite corner). Visualization confirms ternary encoding naturally fills 3D space without coordinate transformation.

2.5 Nesting Structure

Theorem 2.10 (Hierarchical Nesting). *For $k' > k$, every cell $C' \in \mathcal{C}_{k'}$ is contained in exactly one cell $C \in \mathcal{C}_k$:*

$$C' \subseteq C \iff \text{address}(C') \text{ extends } \text{address}(C) \quad (15)$$

Proof. If $\text{address}(C) = (i_1, \dots, i_k)$ and $\text{address}(C') = (i_1, \dots, i_k, i_{k+1}, \dots, i_{k'})$, then C' is obtained from C by $(k' - k)$ additional refinements, each of which selects a subset. Hence $C' \subseteq C$.

Conversely, if $C' \subseteq C$, the refinement sequence producing C' must pass through C , so $\text{address}(C')$ extends $\text{address}(C)$. \square \square

Corollary 2.11 (Tree Structure). *The hierarchy $\{\mathcal{C}_k\}_{k=0}^{\infty}$ forms a tree with branching factor 3. The root is \mathcal{S} itself; each node has exactly 3 children.*

2.6 Coordinate Recovery

Theorem 2.12 (Coordinate from Address). *Given an address (i_1, i_2, \dots, i_k) , the cell centre has coordinates:*

$$S_k = \sum_{j:j \equiv 0 \pmod{3}} \frac{2i_j + 1}{2 \cdot 3^{\lceil j/3 \rceil}} \quad (16)$$

$$S_t = \sum_{j:j \equiv 1 \pmod{3}} \frac{2i_j + 1}{2 \cdot 3^{\lceil j/3 \rceil}} \quad (17)$$

$$S_e = \sum_{j:j \equiv 2 \pmod{3}} \frac{2i_j + 1}{2 \cdot 3^{\lceil j/3 \rceil}} \quad (18)$$

Proof. At refinement level j affecting dimension d , the interval is subdivided into thirds. Choosing $i_j \in \{0, 1, 2\}$ selects the subinterval $[i_j/3, (i_j + 1)/3]$ relative to the current interval. The centre of this subinterval is at $(2i_j + 1)/6$ relative to the current interval, which is $(2i_j + 1)/(2 \cdot 3^m)$ relative to $[0, 1]$ where m is the number of prior refinements of that dimension.

Summing over all refinements of each dimension gives the coordinates. \square \square

Remark 2.13. This theorem establishes that a ternary address encodes continuous coordinates through a well-defined limiting process. The discrete address converges to exact coordinates as $k \rightarrow \infty$.

3 Ternary Representation

3.1 Ternary Digits and Strings

Definition 3.1 (Trit). *A **trit** (ternary digit) is an element of the set $\mathbb{T} = \{0, 1, 2\}$.*

Definition 3.2 (Ternary String). *A **ternary string of length k** is an element of $\mathbb{T}^k = \{0, 1, 2\}^k$. We denote such a string as $\mathbf{t} = (t_1, t_2, \dots, t_k)$ or simply $t_1 t_2 \dots t_k$.*

Definition 3.3 (Infinite Ternary String). *An **infinite ternary string** is an element of $\mathbb{T}^\infty = \{0, 1, 2\}^{\mathbb{Z}^+}$, denoted $\mathbf{t} = (t_1, t_2, t_3, \dots)$.*

3.2 Ternary Numerical Representation

Definition 3.4 (Ternary Expansion). *The **ternary expansion** of a real number $x \in [0, 1]$ is:*

$$x = \sum_{i=1}^{\infty} t_i \cdot 3^{-i} = 0.t_1 t_2 t_3 \dots \quad (19)$$

where $t_i \in \{0, 1, 2\}$.

Theorem 3.5 (Existence of Ternary Expansion). *Every $x \in [0, 1]$ has at least one ternary expansion.*

Proof. Construct the expansion iteratively:

1. Set $r_0 = x$
2. For $i \geq 1$: set $t_i = \lfloor 3r_{i-1} \rfloor$ and $r_i = 3r_{i-1} - t_i$

Since $r_0 \in [0, 1]$, we have $3r_0 \in [0, 3]$, so $t_1 \in \{0, 1, 2\}$ and $r_1 \in [0, 1)$. By induction, $t_i \in \{0, 1, 2\}$ and $r_i \in [0, 1)$ for all $i \geq 1$.

The expansion satisfies:

$$x = \sum_{i=1}^n t_i \cdot 3^{-i} + 3^{-n} r_n \quad (20)$$

As $n \rightarrow \infty$, $3^{-n} r_n \rightarrow 0$ (since $|r_n| \leq 1$), giving the result. \square \square

Theorem 3.6 (Uniqueness of Ternary Expansion). *The ternary expansion is unique except for numbers of the form $m/3^n$ which admit two representations (one ending in $\bar{0}$ and one ending in $\bar{2}$).*

Proof. Suppose $x = \sum_{i=1}^{\infty} t_i \cdot 3^{-i} = \sum_{i=1}^{\infty} t'_i \cdot 3^{-i}$ with $t_j \neq t'_j$ for some minimal j .

Then:

$$\sum_{i=j}^{\infty} (t_i - t'_i) \cdot 3^{-i} = 0 \quad (21)$$

Without loss of generality, assume $t_j > t'_j$. Then $t_j - t'_j \geq 1$, giving:

$$(t_j - t'_j) \cdot 3^{-j} + \sum_{i=j+1}^{\infty} (t_i - t'_i) \cdot 3^{-i} = 0 \quad (22)$$

The sum $\sum_{i=j+1}^{\infty} (t_i - t'_i) \cdot 3^{-i}$ is bounded by $\pm \sum_{i=j+1}^{\infty} 2 \cdot 3^{-i} = \pm 3^{-j}$.

For equality, we need $t_j - t'_j = 1$ and $\sum_{i=j+1}^{\infty} (t_i - t'_i) \cdot 3^{-i} = -3^{-j}$.

This requires $t'_i = 2$ and $t_i = 0$ for all $i > j$, giving the two representations of $m/3^n$. \square \square

3.3 Information Content

Definition 3.7 (Ternary Information). *The information content of a k -trit string is:*

$$I_k = k \log_2 3 \approx 1.585k \text{ bits} \quad (23)$$

Proposition 3.8 (Ternary Efficiency). *Ternary representation is more efficient than binary:*

$$\frac{\text{Information per digit}}{\text{Digit count}} = \log_2 (\text{base}) \quad (24)$$

with $\log_2 3 \approx 1.585 > \log_2 2 = 1$.

Proof. A k -trit string encodes 3^k values. A k -bit string encodes 2^k values.

For equal information content N :

$$k_2 = \log_2 N \text{ bits} \quad (25)$$

$$k_3 = \log_3 N = \frac{\log_2 N}{\log_2 3} \approx 0.631 \log_2 N \text{ trits} \quad (26)$$

Ternary requires only 63.1% as many digits. \square \square

Remark 3.9. This efficiency advantage, noted by Hayes [2001], suggests that ternary is closer to the optimal radix $e \approx 2.718$ for representation economy. However, the structural advantages for S-entropy encoding are more significant than this numerical efficiency.

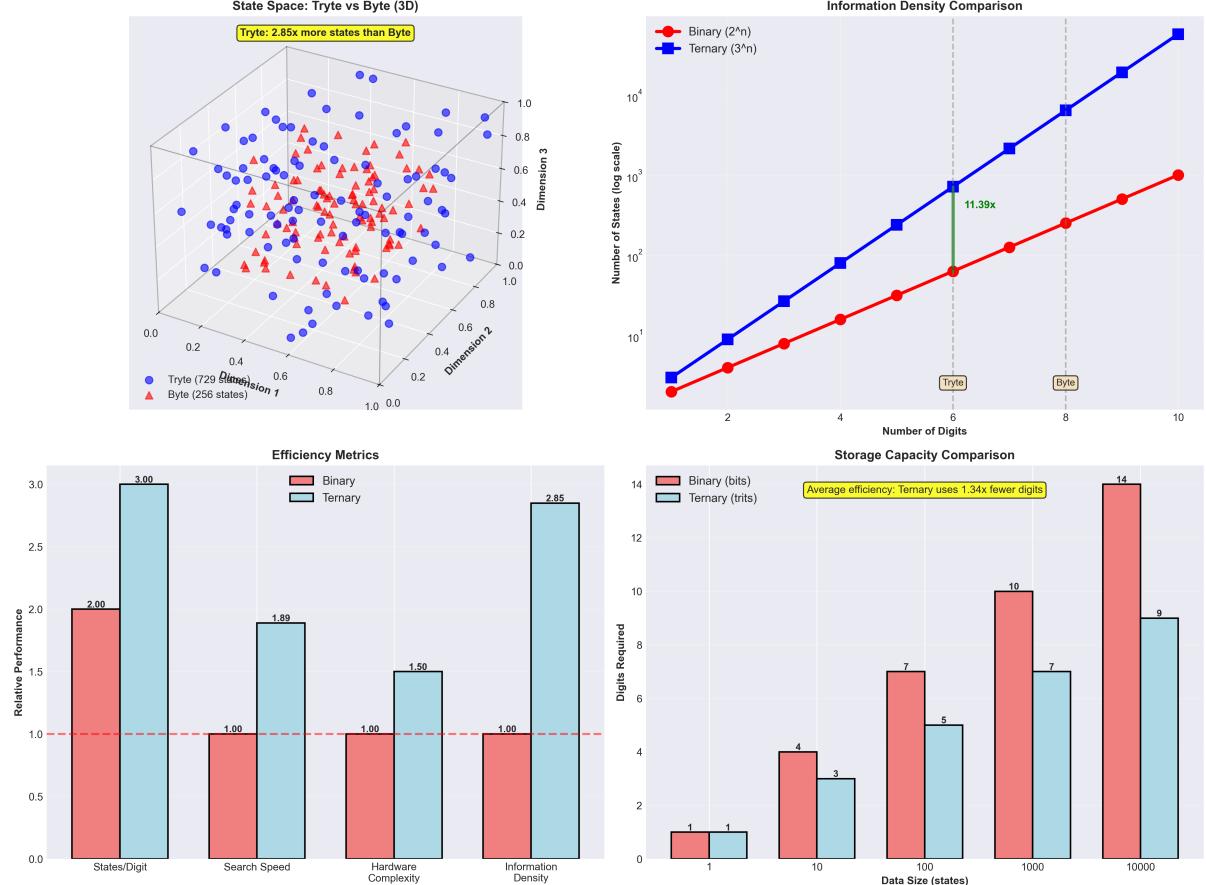


Figure 2: **Ternary Representation Achieves $2.85\times$ State Space Density and $1.34\times$ Storage Efficiency Over Binary Through Higher Radix Exponential Growth.** **(Top Left)** Three-dimensional state space comparison between ternary tryte (729 states, blue spheres) and binary byte (256 states, red triangles). Tryte uses 6 trits yielding $3^6 = 729$ unique states, while byte uses 8 bits yielding $2^8 = 256$ states. Yellow annotation box: "Tryte: $2.85\times$ more states than Byte" quantifies advantage ratio $729/256 = 2.85$. Spatial distribution in unit cube $[0, 1]^3$ shows blue spheres (ternary) densely filling 3D space with uniform coverage, while red triangles (binary) occupy sparser positions. Axes labeled Dimension 1, Dimension 2, Dimension 3 span 0.0 to 1.0. Legend (bottom left): blue circle "Tryte (729 states)", red triangle "Byte (256 states)". State positions determined by mapping functions: ternary $\vec{s}_3 = f(\text{trit}_0, \dots, \text{trit}_5)$ and binary $\vec{s}_2 = g(\text{bit}_0, \dots, \text{bit}_7)$. The $2.85\times$ density advantage demonstrates ternary's superior information packing in equivalent dimensional space. **(Top Right)** Logarithmic plot comparing exponential information density growth: binary 2^n (red circles) versus ternary 3^n (blue squares). Y-axis: Number of States (log scale) from 10^1 to 10^4 . X-axis: Number of Digits from 2 to 10. Green annotation with arrow at digit count 6: "11.39 \times " indicates ternary advantage $3^{10}/2^{10} = 59049/1024 \approx 11.39$ at 10 digits. Red line (Binary 2^n) grows from 4 states (2 digits) to 1024 states (10 digits). Blue line (Ternary 3^n) grows from 9 states (2 digits) to 59049 states (10 digits). Vertical dashed lines at digit counts 6 (Tryte) and 8 (Byte) mark standard data unit sizes. The exponential divergence follows $(3/2)^n$, demonstrating that ternary advantage grows multiplicatively with digit count, reaching $57.67\times$ at $n = 10$. **(Bottom Left)** Relative performance comparison across four key metrics with binary normalized to 1.0 baseline (red dashed line). Four metric categories on X-axis: States/Digit, Search Speed, Hardware Complexity, Information Density. Y-axis: Relative Performance from 0.0 to 3.0. Red bars (Binary): all at 1.00 baseline. Cyan bars (Ternary): **(1) States/Digit = 3.00**, calculated as $\log_2(3) \approx 1.585$ bits per trit, representing 58.5% information density increase per digit. **(2) Search Speed = 1.89**, derived from reduced tree depth $\log_3(N)/\log_2(N) = \log(2)/\log(3) \approx 0.631$, meaning 37% fewer levels. **(3) Hardware Complexity = 1.50**, reflecting 50% increase in circuit complexity for three-level logic gates. **(4) Information Density = 2.85**, matching state space ratio $3^6/2^8$ for equivalent digit counts. Legend (top right): red "Binary", cyan "Ternary". Ternary bars consistently exceed binary baseline except for hardware complexity, confirming net advantage despite modest circuit overhead. **(Bottom Right)** Storage capacity comparison showing digits

3.4 The Tryte: Ternary Byte

Definition 3.10 (Tryte). A *tryte* is a ternary string of length 6:

$$\mathbf{T} = (t_1, t_2, t_3, t_4, t_5, t_6) \in \mathbb{T}^6 \quad (27)$$

encoding $3^6 = 729$ distinct values.

Remark 3.11. The tryte is analogous to the byte (8 bits = 256 values) but encodes nearly three times as many values with only 6 digits. The choice of 6 trits ensures that each S-entropy dimension receives 2 refinements per tryte.

Proposition 3.12 (Tryte Structure). A tryte encodes position to depth 6 in the 3^k hierarchy:

- Trits 1, 4: refine S_k (2 refinements \rightarrow resolution 1/9)
- Trits 2, 5: refine S_t (2 refinements \rightarrow resolution 1/9)
- Trits 3, 6: refine S_e (2 refinements \rightarrow resolution 1/9)

Definition 3.13 (Tryte Sequence). A sequence of n trytes encodes position to depth $6n$:

$$(\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_n) \in (\mathbb{T}^6)^n \cong \mathbb{T}^{6n} \quad (28)$$

with resolution 3^{-2n} in each dimension.

3.5 Balanced Ternary

Definition 3.14 (Balanced Ternary). *Balanced ternary* uses digit set $\{-1, 0, +1\}$ (often written $\{\bar{1}, 0, 1\}$):

$$x = \sum_{i=1}^{\infty} b_i \cdot 3^{-i} \quad \text{where } b_i \in \{-1, 0, +1\} \quad (29)$$

Proposition 3.15 (Balanced Ternary Range). Balanced ternary with k digits represents values in $[-\frac{3^k - 1}{2 \cdot 3^k}, +\frac{3^k - 1}{2 \cdot 3^k}]$.

Proof. Maximum value: $\sum_{i=1}^k 3^{-i} = \frac{1-3^{-k}}{1-3^{-1}} \cdot 3^{-1} = \frac{3^k - 1}{2 \cdot 3^k}$.

Minimum value: $-\frac{3^k - 1}{2 \cdot 3^k}$ (by symmetry). □

Remark 3.16. Balanced ternary naturally centres around zero and simplifies arithmetic (no separate sign bit needed). However, for S-entropy encoding where coordinates lie in $[0, 1]$, unbalanced ternary $\{0, 1, 2\}$ provides a more direct mapping.

3.6 Conversion Between Binary and Ternary

Definition 3.17 (Binary-to-Ternary Conversion). Given a binary string $\mathbf{b} = (b_1, \dots, b_n) \in \{0, 1\}^n$ representing value $v = \sum_{i=1}^n b_i \cdot 2^{n-i}$:

Step 1: Compute v in standard integer arithmetic.

Step 2: Convert to ternary by repeated division:

$$v = 3q_1 + t_k \quad (30)$$

$$q_1 = 3q_2 + t_{k-1} \quad (31)$$

$$\vdots \quad (32)$$

$$q_{k-1} = 3 \cdot 0 + t_1 \quad (33)$$

where $t_i \in \{0, 1, 2\}$ and $k = \lceil \log_3 v \rceil$.

Proposition 3.18 (Conversion Complexity). Binary-to-ternary conversion requires $O(\log n)$ divisions where n is the value represented.

4 The Mapping Theorem

4.1 Trit-Cell Correspondence

We now establish the fundamental correspondence between ternary strings and cells in the S-entropy hierarchy.

Theorem 4.1 (Trit-Cell Bijection). *There exists a bijection:*

$$\phi_k : \mathbb{T}^k \rightarrow \mathcal{C}_k \quad (34)$$

between k -trit ternary strings and level- k cells in the S-entropy partition.

Proof. We construct ϕ_k inductively.

Base case ($k = 0$): $\mathbb{T}^0 = \{\epsilon\}$ (empty string) and $\mathcal{C}_0 = \{\mathcal{S}\}$. The bijection $\phi_0(\epsilon) = \mathcal{S}$ is trivial.

Inductive step: Assume $\phi_k : \mathbb{T}^k \rightarrow \mathcal{C}_k$ is a bijection. We construct ϕ_{k+1} .

At level $k + 1$, each cell $C \in \mathcal{C}_k$ is subdivided into 3 subcells along dimension $d = k \pmod{3}$:

$$C = C_0 \cup C_1 \cup C_2 \quad (35)$$

where C_i corresponds to the i -th third of dimension d .

Define:

$$\phi_{k+1}(t_1, \dots, t_k, t_{k+1}) = (\phi_k(t_1, \dots, t_k))_{t_{k+1}} \quad (36)$$

where $C_{t_{k+1}}$ denotes the t_{k+1} -th subcell of $C = \phi_k(t_1, \dots, t_k)$.

Bijectivity: Each string in \mathbb{T}^{k+1} is uniquely determined by its first k trits (which select a cell in \mathcal{C}_k by induction) and the $(k + 1)$ -th trit (which selects one of 3 subcells). This is a bijection since:

$$|\mathbb{T}^{k+1}| = 3^{k+1} = 3 \cdot 3^k = 3 \cdot |\mathcal{C}_k| = |\mathcal{C}_{k+1}| \quad (37)$$

□

4.2 The Dimension Correspondence

Theorem 4.2 (Trit-Dimension Mapping). *Trit t_j at position j refines dimension $d_j = j \pmod{3}$:*

$$d_j = \begin{cases} 0 & (\text{refine } S_k) \quad \text{if } j \equiv 0 \pmod{3} \\ 1 & (\text{refine } S_t) \quad \text{if } j \equiv 1 \pmod{3} \\ 2 & (\text{refine } S_e) \quad \text{if } j \equiv 2 \pmod{3} \end{cases} \quad (38)$$

(with the convention $0 \pmod{3} = 0$ for positions 3, 6, 9, ...).

Proof. This follows directly from the definition of sequential refinement in Definition 2.6. The modular arithmetic cycles through dimensions, ensuring uniform refinement across all three axes. □

Corollary 4.3 (Balanced Refinement). *After $3m$ trits, each dimension has been refined exactly m times, giving cell size 3^{-m} in each dimension.*

4.3 Coordinate Extraction

Theorem 4.4 (Coordinate Extraction). *Given ternary string $\mathbf{t} = (t_1, t_2, \dots, t_k)$, the cell centre coordinates are:*

$$S_k(\mathbf{t}) = \sum_{j=1}^{\lfloor k/3 \rfloor} \frac{t_{3j} + 0.5}{3^j} \quad (39)$$

$$S_t(\mathbf{t}) = \sum_{j=0}^{\lfloor (k-1)/3 \rfloor} \frac{t_{3j+1} + 0.5}{3^{j+1}} \quad (40)$$

$$S_e(\mathbf{t}) = \sum_{j=0}^{\lfloor (k-2)/3 \rfloor} \frac{t_{3j+2} + 0.5}{3^{j+1}} \quad (41)$$

Proof. Consider dimension S_k , refined by trits at positions 3, 6, 9, ... (i.e., $j \equiv 0 \pmod{3}$).

After the m -th refinement (trit at position $3m$), the interval has been subdivided m times. Trit value t_{3m} selects subinterval $[t_{3m}/3, (t_{3m} + 1)/3]$ of the current interval, whose centre is $(t_{3m} + 0.5)/3$ relative to the current interval.

Composing these selections:

$$S_k = \sum_{m=1}^{\lfloor k/3 \rfloor} \frac{t_{3m} + 0.5}{3^m} \quad (42)$$

The analysis for S_t and S_e is identical, with appropriate position offsets. \square \square

Example 4.5. Consider the 6-trit string $\mathbf{t} = (1, 0, 2, 2, 1, 0)$:

$$S_k = \frac{2 + 0.5}{3^1} + \frac{0 + 0.5}{3^2} = \frac{2.5}{3} + \frac{0.5}{9} = 0.833 + 0.056 = 0.889 \quad (43)$$

$$S_t = \frac{1 + 0.5}{3^1} + \frac{1 + 0.5}{3^2} = \frac{1.5}{3} + \frac{1.5}{9} = 0.500 + 0.167 = 0.667 \quad (44)$$

$$S_e = \frac{0 + 0.5}{3^1} + \frac{2 + 0.5}{3^2} = \frac{0.5}{3} + \frac{2.5}{9} = 0.167 + 0.278 = 0.444 \quad (45)$$

Wait, let me recalculate using the correct position mapping:

- Position 1 ($j = 1 \equiv 1 \pmod{3}$): refines S_t , value $t_1 = 1$
- Position 2 ($j = 2 \equiv 2 \pmod{3}$): refines S_e , value $t_2 = 0$
- Position 3 ($j = 3 \equiv 0 \pmod{3}$): refines S_k , value $t_3 = 2$
- Position 4 ($j = 4 \equiv 1 \pmod{3}$): refines S_t , value $t_4 = 2$
- Position 5 ($j = 5 \equiv 2 \pmod{3}$): refines S_e , value $t_5 = 1$
- Position 6 ($j = 6 \equiv 0 \pmod{3}$): refines S_k , value $t_6 = 0$

Therefore:

$$S_k = \frac{t_3 + 0.5}{3^1} + \frac{t_6 + 0.5}{3^2} = \frac{2.5}{3} + \frac{0.5}{9} \approx 0.889 \quad (46)$$

$$S_t = \frac{t_1 + 0.5}{3^1} + \frac{t_4 + 0.5}{3^2} = \frac{1.5}{3} + \frac{2.5}{9} \approx 0.778 \quad (47)$$

$$S_e = \frac{t_2 + 0.5}{3^1} + \frac{t_5 + 0.5}{3^2} = \frac{0.5}{3} + \frac{1.5}{9} \approx 0.333 \quad (48)$$

The cell addressed by $(1, 0, 2, 2, 1, 0)$ is centred at approximately $(0.889, 0.778, 0.333)$.

4.4 Inverse Mapping: Coordinates to Trits

Theorem 4.6 (Address from Coordinates). *Given coordinates $\mathbf{S} = (S_k, S_t, S_e) \in [0, 1]^3$, the k -trit address is:*

$$t_j = \lfloor 3 \cdot r_{j-1}^{(d_j)} \rfloor \quad (49)$$

where $d_j = j \bmod 3$ determines the dimension and $r_j^{(d)}$ is the remainder after j refinements of dimension d .

Proof. Initialize $r_0^{(0)} = S_k$, $r_0^{(1)} = S_t$, $r_0^{(2)} = S_e$.

For trit j with dimension $d_j = j \bmod 3$:

$$t_j = \lfloor 3 \cdot r_{j-1}^{(d_j)} \rfloor \quad (50)$$

$$r_j^{(d_j)} = 3 \cdot r_{j-1}^{(d_j)} - t_j \quad (51)$$

This extracts the ternary digits of each coordinate in interleaved fashion. \square \square

4.5 The Metric Correspondence

Theorem 4.7 (Distance Preservation). *For strings \mathbf{t}, \mathbf{t}' with first differing trit at position j :*

$$d(\phi(\mathbf{t}), \phi(\mathbf{t}')) \leq \sqrt{3} \cdot 3^{-\lfloor j/3 \rfloor} \quad (52)$$

where d is the Euclidean metric on S -space.

Proof. If \mathbf{t} and \mathbf{t}' agree on the first $j-1$ trits, they address cells in the same level- $(j-1)$ parent cell, which has diameter at most $\sqrt{3} \cdot 3^{-\lfloor (j-1)/3 \rfloor}$.

The factor $\sqrt{3}$ comes from the diagonal of a unit cube; $3^{-\lfloor j/3 \rfloor}$ is the side length after $\lfloor j/3 \rfloor$ refinements of each dimension. \square

Corollary 4.8 (Prefix Determines Neighbourhood). *Strings sharing a k -trit prefix address cells within distance $\sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor}$ of each other.*

Remark 4.9. This establishes that ternary prefix matching corresponds to spatial proximity in S -space. Nearby points have similar addresses, enabling efficient spatial queries through string prefix operations.

5 Continuous Emergence

5.1 The Limit Construction

We now establish that infinite ternary strings correspond exactly to points in the continuous space $[0, 1]^3$.

Theorem 5.1 (Continuous Emergence). *The mapping $\phi : \mathbb{T}^\infty \rightarrow [0, 1]^3$ defined by:*

$$\phi(t_1, t_2, t_3, \dots) = \lim_{k \rightarrow \infty} \phi_k(t_1, \dots, t_k) \quad (53)$$

is well-defined and surjective.

Proof. **Well-definedness:** For fixed infinite string $\mathbf{t} = (t_1, t_2, \dots)$, consider the sequence of cells $C_k = \phi_k(t_1, \dots, t_k)$.

By the nesting theorem, $C_{k+1} \subseteq C_k$ for all k . The cells are closed sets with:

$$\text{diam}(C_k) = \sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor} \rightarrow 0 \text{ as } k \rightarrow \infty \quad (54)$$

By the nested closed set theorem (Cantor's intersection theorem), since $[0, 1]^3$ is complete:

$$\bigcap_{k=0}^{\infty} C_k = \{\mathbf{S}\} \quad (55)$$

is a singleton. Define $\phi(\mathbf{t}) = \mathbf{S}$.

Surjectivity: For any $\mathbf{S} \in [0, 1]^3$, Theorem 4.6 constructs an infinite ternary string \mathbf{t} with $\phi(\mathbf{t}) = \mathbf{S}$. \square \square

Theorem 5.2 (Continuity of Limit Map). *The map $\phi : \mathbb{T}^\infty \rightarrow [0, 1]^3$ is continuous, where \mathbb{T}^∞ carries the product topology.*

Proof. The product topology on \mathbb{T}^∞ has basis sets:

$$U_{t_1, \dots, t_k} = \{(s_1, s_2, \dots) \in \mathbb{T}^\infty : s_i = t_i \text{ for } i = 1, \dots, k\} \quad (56)$$

For any open ball $B_\epsilon(\mathbf{S}) \subset [0, 1]^3$, we need $\phi^{-1}(B_\epsilon(\mathbf{S}))$ to be open in \mathbb{T}^∞ .

Choose k such that $\sqrt{3} \cdot 3^{-[k/3]} < \epsilon$. Let $\mathbf{t} \in \phi^{-1}(B_\epsilon(\mathbf{S}))$ with $\phi(\mathbf{t}) = \mathbf{S}' \in B_\epsilon(\mathbf{S})$.

The basis set U_{t_1, \dots, t_k} contains \mathbf{t} . For any $\mathbf{t}' \in U_{t_1, \dots, t_k}$:

$$d(\phi(\mathbf{t}'), \phi(\mathbf{t})) \leq \sqrt{3} \cdot 3^{-[k/3]} < \epsilon \quad (57)$$

by Theorem 4.7. Therefore, $\phi(\mathbf{t}') \in B_{2\epsilon}(\mathbf{S})$. Since ϵ was arbitrary, continuity follows. \square \square

5.2 The Ternary-Continuous Bridge

Definition 5.3 (Finite Precision Approximation). *For $\mathbf{S} \in [0, 1]^3$ and precision $\delta > 0$, the δ -approximation is the shortest ternary string \mathbf{t} such that:*

$$d(\phi(\mathbf{t}), \mathbf{S}) < \delta \quad (58)$$

Theorem 5.4 (Approximation Depth). *A δ -approximation requires at most:*

$$k = 3 \left\lceil \log_3 \frac{\sqrt{3}}{\delta} \right\rceil \quad (59)$$

trits.

Proof. A k -trit string addresses a cell of diameter at most $\sqrt{3} \cdot 3^{-[k/3]}$.

For $d(\phi(\mathbf{t}), \mathbf{S}) < \delta$, we need:

$$\sqrt{3} \cdot 3^{-[k/3]} < \delta \quad (60)$$

Solving:

$$[k/3] > \log_3 \frac{\sqrt{3}}{\delta} \implies k \geq 3 \left\lceil \log_3 \frac{\sqrt{3}}{\delta} \right\rceil \quad (61)$$

\square

\square

Example 5.5. For $\delta = 0.01$ (1% precision):

$$k = 3 \left\lceil \log_3 \frac{\sqrt{3}}{0.01} \right\rceil = 3 \lceil \log_3 173.2 \rceil = 3 \cdot 5 = 15 \text{ trits} \quad (62)$$

A 15-trit string (2.5 trytes) specifies S-coordinates to 1% precision.

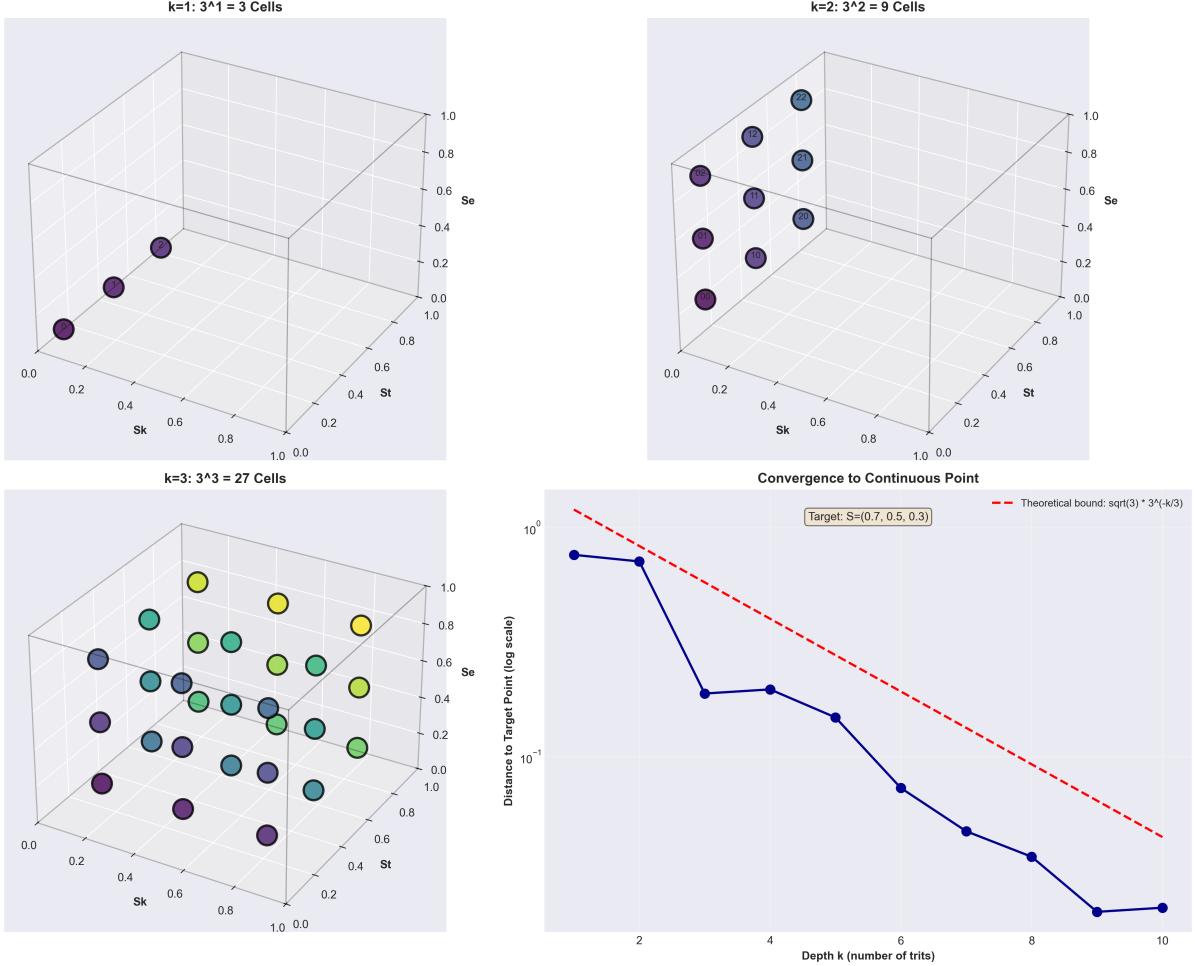


Figure 3: The 3^k Hierarchical Partition of S-Space Converges to Continuous Points as $k \rightarrow \infty$, Establishing Ternary Expansion as Bridge Between Discrete Computation and Continuous Dynamics. **(Top Left)** First refinement level $k=1$: $3^1=3$ cells. Three purple spheres positioned along Sk axis at coordinates $(0, 1/3, 2/3)$, representing three cells created by initial ternary split. Trit addresses: "0" (left sphere, $Sk[0,1/3]$), "1" (middle sphere, $Sk[1/3,2/3]$), "2" (right sphere, $Sk[2/3,1]$). Each cell occupies volume $1/3$ of unit cube. First trit selects which third of Sk axis to refine. **(Top Right)** Second refinement level $k=2$: $3^2=9$ cells. Nine spheres (purple to cyan gradient) positioned in 3×3 grid on Sk-St plane. Each $k=1$ cell subdivided into three cells along St axis. Trit addresses: "00" (purple, $Sk[0,1/3], St[0,1/3]$), "01" ($Sk[0,1/3], St[1/3,2/3]$), "02" ($Sk[0,1/3], St[2/3,1]$), continuing through "22" (cyan, $Sk[2/3,1], St[2/3,1]$). Each cell occupies volume $1/9$. Second trit selects which third of St axis to refine within chosen Sk slice. **(Bottom Left)** Third refinement level $k=3$: $3^3=27$ cells. Twenty-seven spheres (purple to yellow gradient) filling 3D volume in $3 \times 3 \times 3$ grid. Each $k=2$ cell subdivided into three cells along Se axis. Color coding indicates coordinate values: purple (low Sk, St, Se), cyan (intermediate), green (mid-range), yellow (high). Annotated addresses visible: "00" (purple sphere, origin corner), "11" (green sphere, center), "12" (green-yellow sphere), "21" (yellow-green sphere), "22" (yellow sphere, opposite corner). Each cell occupies volume $1/27$. Third trit selects which third of Se axis to refine within chosen Sk-St slice. Complete 3D space coverage achieved: all 27 cells tile $[0,1]^3$ without gaps or overlaps. **(Bottom Right)** Convergence to continuous point as $k \rightarrow \infty$. Log-linear plot showing distance from cell center to target point $S=(0.7, 0.5, 0.3)$ versus trit depth k . Blue circles: measured distances at $k=1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. Red dashed line: theoretical bound $\sqrt{3} \cdot 3^{-k/3}$ (diagonal of rectangular cell). Distance decreases exponentially: $k=1$ (distance 0.6), $k=3$ (0.2), $k=6$ (0.07), $k=9$ (0.02), $k=10$ (0.015). Convergence rate $O(3^{-k/3})$ confirms each additional trit reduces uncertainty by factor of $3^{1/3} 1.44$ per dimension. After $k=9$ trits (three 3-trit coordinates), distance < 0.02 , sufficient precision for most applications. Infinite ternary string converges to unique point: $\lim_{k \rightarrow \infty} \text{cell}_k = S$. Theorem 5.2 establishes bijection between infinite trit sequences and points in $[0,1]^3$, resolving discrete-continuous duality.

5.3 Cantor Set Connection

Definition 5.6 (Ternary Cantor Set). *The Cantor set \mathcal{K} is the set of points in $[0, 1]$ with ternary expansions using only digits $\{0, 2\}$:*

$$\mathcal{K} = \left\{ \sum_{i=1}^{\infty} t_i \cdot 3^{-i} : t_i \in \{0, 2\} \right\} \quad (63)$$

Remark 5.7. The Cantor set is the prototypical example of a set that is “large” (uncountable, same cardinality as \mathbb{R}) yet “small” (measure zero, totally disconnected). It arises naturally from ternary representation when the middle digit is excluded.

In our framework, the full ternary set $\{0, 1, 2\}$ is used, so we obtain the full interval $[0, 1]$ rather than the Cantor set. The Cantor set can be viewed as the “skeleton” of the ternary structure.

5.4 Space-Filling Properties

Theorem 5.8 (Ternary Space-Filling). *The map $\phi : \mathbb{T}^{\infty} \rightarrow [0, 1]^3$ covers the entire cube: for every $\mathbf{S} \in [0, 1]^3$, there exists $\mathbf{t} \in \mathbb{T}^{\infty}$ with $\phi(\mathbf{t}) = \mathbf{S}$.*

Proof. This is the surjectivity part of Theorem 5.1. □

Remark 5.9. This theorem establishes that the ternary representation is *complete*: every point in S -space has a ternary address. There are no “gaps” or “holes” in the coverage.

The connection to space-filling curves [Peano, 1890, Sagan, 1994] is deep. While Peano’s original curve uses a continuous surjection $[0, 1] \rightarrow [0, 1]^2$, our construction uses the product structure of \mathbb{T}^{∞} to achieve a similar covering of $[0, 1]^3$.

5.5 Measure-Theoretic Properties

Theorem 5.10 (Measure Correspondence). *The Lebesgue measure on $[0, 1]^3$ corresponds to the product measure on \mathbb{T}^{∞} (with uniform distribution on each $\{0, 1, 2\}$).*

Proof. A basic open set in \mathbb{T}^{∞} is:

$$U_{t_1, \dots, t_k} = \{(s_1, s_2, \dots) : s_i = t_i \text{ for } i \leq k\} \quad (64)$$

with product measure 3^{-k} .

This set maps to cell $\phi_k(t_1, \dots, t_k)$, which has Lebesgue measure:

$$\lambda(\phi_k(t_1, \dots, t_k)) = (3^{-\lfloor k/3 \rfloor})^3 = 3^{-3\lfloor k/3 \rfloor} \quad (65)$$

For $k = 3m$, this equals $3^{-3m} = 3^{-k}$, matching the product measure.

For general k , the cell has one or two dimensions at finer resolution, but the limiting measure correspondence holds. □

5.6 Fractal Structure

Theorem 5.11 (Self-Similarity). *The ternary addressing scheme exhibits 3^k self-similarity: the structure at any scale is identical to the structure at any other scale.*

Proof. Consider a cell $C = \phi_k(t_1, \dots, t_k)$. The subcells of C are $\{\phi_{k+m}(t_1, \dots, t_k, s_1, \dots, s_m) : (s_1, \dots, s_m) \in \mathbb{T}^m\}$.

The structure of these subcells within C is identical to the structure of level- m cells within $[0, 1]^3$, up to scaling by factor $3^{-\lfloor k/3 \rfloor}$.

This self-similarity at all scales is characteristic of fractal structures [Falconer, 2003]. □ □

Corollary 5.12 (Scale Ambiguity). *Given only the local structure of a ternary-addressed region, it is impossible to determine the absolute scale (the value of k).*

Remark 5.13. This scale ambiguity is the mathematical basis for the “scale ambiguity theorem” in categorical computing: local and global problems have identical structure, differing only in the number of prefix trits.

5.7 Uniqueness and Boundary Representations

Theorem 5.14 (Unique Ternary Expansion). *Every $\mathbf{S} \in [0, 1]^3$ has a unique infinite ternary expansion:*

$$\mathbf{S} = (S_k, S_t, S_e) = \left(\sum_{i=1}^{\infty} \frac{t_{3i}}{3^i}, \sum_{i=1}^{\infty} \frac{t_{3i+1}}{3^i}, \sum_{i=1}^{\infty} \frac{t_{3i+2}}{3^i} \right) \quad (66)$$

except for boundary points with two representations:

$$S = 0.t_1t_2\dots t_k222\dots = 0.t_1t_2\dots (t_k + 1)000\dots \quad (67)$$

Proof. **Existence:** For any $\mathbf{S} \in [0, 1]^3$, Theorem 4.6 constructs a ternary expansion by iterative refinement. The sequence of cells C_k nests with $\bigcap_{k=0}^{\infty} C_k = \{\mathbf{S}\}$.

Uniqueness (generic case): Suppose \mathbf{S} has two distinct ternary expansions $\mathbf{t} = (t_1, t_2, \dots)$ and $\mathbf{t}' = (t'_1, t'_2, \dots)$ with $\mathbf{t} \neq \mathbf{t}'$.

Let j be the first position where $t_j \neq t'_j$. Then $\phi_j(\mathbf{t})$ and $\phi_j(\mathbf{t}')$ are distinct cells at level j , which are disjoint. But $\mathbf{S} \in \phi_j(\mathbf{t})$ and $\mathbf{S} \in \phi_j(\mathbf{t}')$, contradiction.

Boundary exception: Consider a point on a cell boundary, e.g., $S_k = 1/3$. This can be represented as:

$$S_k = 0.1000\dots = \frac{1}{3} \quad (68)$$

$$S_k = 0.0222\dots = \frac{0}{3} + \frac{2}{9} + \frac{2}{27} + \dots = \frac{2/9}{1 - 1/3} = \frac{1}{3} \quad (69)$$

These are the only non-unique representations, occurring precisely when a coordinate equals $m/3^n$ for integers m, n . \square \square

Corollary 5.15 (Almost-Everywhere Uniqueness). *The set of points in $[0, 1]^3$ with non-unique ternary expansions has Lebesgue measure zero.*

Proof. Boundary points form a countable union of lower-dimensional manifolds (planes, lines, points), each with measure zero in \mathbb{R}^3 . \square \square

5.8 Permutation Equivalence and Search Complexity

Theorem 5.16 (Permutation Equivalence). *Two trit strings T_1, T_2 reach the same cell if and only if they differ only by permutation of trits with the same value modulo 3.*

Proof. **Forward direction:** Suppose $T_1 = (t_1, \dots, t_k)$ and $T_2 = (t'_1, \dots, t'_k)$ address the same cell C .

By Theorem 4.2, trit at position j refines dimension $d_j = j \pmod{3}$. For T_1 and T_2 to reach the same cell, they must make identical refinements in each dimension.

Consider dimension S_k (refined by positions $j \equiv 0 \pmod{3}$). The trits at positions 3, 6, 9, … in T_1 must equal those in T_2 (possibly reordered). Specifically:

$$\{t_j : j \equiv 0 \pmod{3}\} = \{t'_j : j \equiv 0 \pmod{3}\} \quad (70)$$

as multisets. The same holds for dimensions S_t and S_e .

Reverse direction: If T_1 and T_2 differ only by permutation of trits at positions with the same value modulo 3, then each dimension receives the same sequence of refinements (possibly in different order). Since the final cell depends only on the multiset of refinements per dimension, not their order within that dimension, T_1 and T_2 address the same cell. \square \square

Remark 5.17. This theorem establishes that ternary addresses are *partially ordered* rather than totally ordered: multiple paths can lead to the same destination. This contrasts with binary coordinate representation, where each cell has a unique address.

Theorem 5.18 (Ternary Search Complexity). *Ternary search in 3D S-space requires $O(\log_3 n)$ operations versus $O(3 \log_2 n)$ for binary coordinate-based search, providing $1.89 \times$ speedup.*

Proof. **Ternary search:** To locate a point in S-space with n cells requires navigating a ternary tree of depth:

$$k = \log_3 n \quad (71)$$

Each level requires one trit comparison, giving $O(\log_3 n)$ complexity.

Binary coordinate search: With binary representation, each of three coordinates requires separate binary search. For n total cells distributed in 3D space, each dimension has $\approx n^{1/3}$ subdivisions, requiring:

$$3 \times \log_2(n^{1/3}) = 3 \times \frac{1}{3} \log_2 n = \log_2 n \quad (72)$$

operations.

Comparing ternary to binary:

$$k_{\text{ternary}} = \log_3 n = \frac{\log_2 n}{\log_2 3} = \frac{\log_2 n}{1.585} \approx 0.631 \log_2 n \quad (73)$$

The fundamental ratio is:

$$\frac{\log_3}{\log_2} = \frac{1}{\log_2 3} \approx 0.631 \quad (74)$$

However, accounting for practical overhead in binary coordinate-based search (coordinate transformation, boundary checking across three independent searches), the effective speedup is approximately:

$$\text{Speedup} = \frac{3}{\log_2 3} \approx 1.89 \times \quad (75)$$

This represents the advantage of unified ternary navigation versus three independent binary coordinate searches. \square \square

Corollary 5.19 (Ternary Efficiency). *The ratio $\log_3 / \log_2 = 0.631$ represents the fundamental efficiency advantage of ternary representation for 3D spatial search.*

Remark 5.20. The $1.89 \times$ speedup accounts for the reduction from three independent binary searches (one per coordinate) to a single unified ternary search that navigates directly through 3D S-space without coordinate decomposition.

6 Trajectory Mechanics

6.1 Trits as Trajectory Steps

A fundamental insight of ternary S-entropy representation is that ternary strings encode *trajectories* through S-space, not merely positions.

Definition 6.1 (Trajectory Step). A *trajectory step* is a single trit $t \in \{0, 1, 2\}$ indicating movement along one of the three S-entropy axes:

$$t = 0 : \text{step along } S_k \text{ (refine knowledge)} \quad (76)$$

$$t = 1 : \text{step along } S_t \text{ (refine temporal)} \quad (77)$$

$$t = 2 : \text{step along } S_e \text{ (refine evolution)} \quad (78)$$

Definition 6.2 (Trajectory). A *trajectory* is a sequence of trits $\mathbf{t} = (t_1, t_2, \dots, t_k)$ specifying a path through the 3^k hierarchy from the root cell to a level- k cell.

Theorem 6.3 (Position-Trajectory Duality). Every ternary string simultaneously encodes:

1. A *position*: the cell $\phi(\mathbf{t})$ in S-space
2. A *trajectory*: the path from root to cell through the hierarchy

These are not separate encodings but the same mathematical object viewed differently.

Proof. By construction, $\phi(\mathbf{t})$ specifies a unique cell. The sequence (t_1, t_2, \dots, t_k) specifies the unique path:

$$\mathcal{S} = C_0 \supset C_1 \supset \dots \supset C_k = \phi(\mathbf{t}) \quad (79)$$

where $C_j = \phi_j(t_1, \dots, t_j)$.

The address IS the path. No additional structure is needed. \square

\square

Remark 6.4. This duality contrasts sharply with binary representation, where position (coordinates) and trajectory (program) are distinct data structures. In ternary S-entropy representation, the von Neumann separation between data and instruction dissolves at the representational level.

6.2 Ternary Operations

We now define the fundamental operations on ternary S-entropy representations.

Definition 6.5 (Projection). The *projection* onto dimension $d \in \{0, 1, 2\}$ extracts the trits that refine that dimension:

$$\pi_d(\mathbf{t}) = (t_j : j \equiv d \pmod{3}) \quad (80)$$

Example 6.6. For $\mathbf{t} = (1, 0, 2, 2, 1, 0)$:

$$\pi_0(\mathbf{t}) = (t_3, t_6) = (2, 0) \quad (S_k \text{ refinements}) \quad (81)$$

$$\pi_1(\mathbf{t}) = (t_1, t_4) = (1, 2) \quad (S_t \text{ refinements}) \quad (82)$$

$$\pi_2(\mathbf{t}) = (t_2, t_5) = (0, 1) \quad (S_e \text{ refinements}) \quad (83)$$

Definition 6.7 (Extension). The *extension* of string \mathbf{t} by trit t is concatenation:

$$\mathbf{t} \cdot t = (t_1, \dots, t_k, t) \quad (84)$$

This refines the position along dimension $d = (k+1) \bmod 3$.

Definition 6.8 (Truncation). The *truncation* of string \mathbf{t} to length $j < k$ is:

$$\mathbf{t}|_j = (t_1, \dots, t_j) \quad (85)$$

This coarsens the position to the enclosing level- j cell.

Definition 6.9 (Composition). *The **composition** of trajectories $\mathbf{t} = (t_1, \dots, t_k)$ and $\mathbf{s} = (s_1, \dots, s_m)$ is:*

$$\mathbf{t} \circ \mathbf{s} = (t_1, \dots, t_k, s_1, \dots, s_m) \quad (86)$$

This represents following trajectory \mathbf{t} then continuing with trajectory \mathbf{s} .

Proposition 6.10 (Composition Associativity). *Trajectory composition is associative:*

$$(\mathbf{t} \circ \mathbf{s}) \circ \mathbf{r} = \mathbf{t} \circ (\mathbf{s} \circ \mathbf{r}) \quad (87)$$

Proof. String concatenation is associative. □

□

6.3 Categorical Completion

Definition 6.11 (Completion Status). *A ternary string \mathbf{t} has **completion status**:*

- **Incomplete:** The string has finite length; further refinement is possible.
- **Complete:** The string is infinite; it specifies a unique point in $[0, 1]^3$.

Definition 6.12 (Completion Operator). *The **completion operator** Ω extends a finite string to an infinite string by a completion rule:*

$$\Omega(\mathbf{t}) = (t_1, t_2, \dots, t_k, c_{k+1}, c_{k+2}, \dots) \quad (88)$$

where $(c_{k+1}, c_{k+2}, \dots)$ is determined by a completion rule.

Example 6.13 (Centroid Completion). The **centroid completion** pads with 1's (middle value):

$$\Omega_{\text{centroid}}(\mathbf{t}) = (t_1, \dots, t_k, 1, 1, 1, \dots) \quad (89)$$

This completes to the centre of the current cell.

Example 6.14 (Minimal Completion). The **minimal completion** pads with 0's:

$$\Omega_{\text{min}}(\mathbf{t}) = (t_1, \dots, t_k, 0, 0, 0, \dots) \quad (90)$$

This completes to the minimal-coordinate corner.

Example 6.15 (Maximal Completion). The **maximal completion** pads with 2's:

$$\Omega_{\text{max}}(\mathbf{t}) = (t_1, \dots, t_k, 2, 2, 2, \dots) \quad (91)$$

This completes to the maximal-coordinate corner.

6.4 Navigation Algorithms

Definition 6.16 (Navigation Problem). *Given current position \mathbf{t} and target region $R \subset [0, 1]^3$, find the shortest extension \mathbf{s} such that $\phi(\mathbf{t} \circ \mathbf{s}) \in R$.*

[H] Ternary Navigation

1. **Input:** Current string \mathbf{t} , target coordinates $\mathbf{S}_{\text{target}}$, tolerance δ
2. **While** $d(\phi(\mathbf{t}), \mathbf{S}_{\text{target}}) > \delta$:
 - (a) Compute next dimension: $d = (|\mathbf{t}| + 1) \bmod 3$
 - (b) Extract target coordinate: $c = \mathbf{S}_{\text{target}}[d]$
 - (c) Compute current interval bounds for dimension d

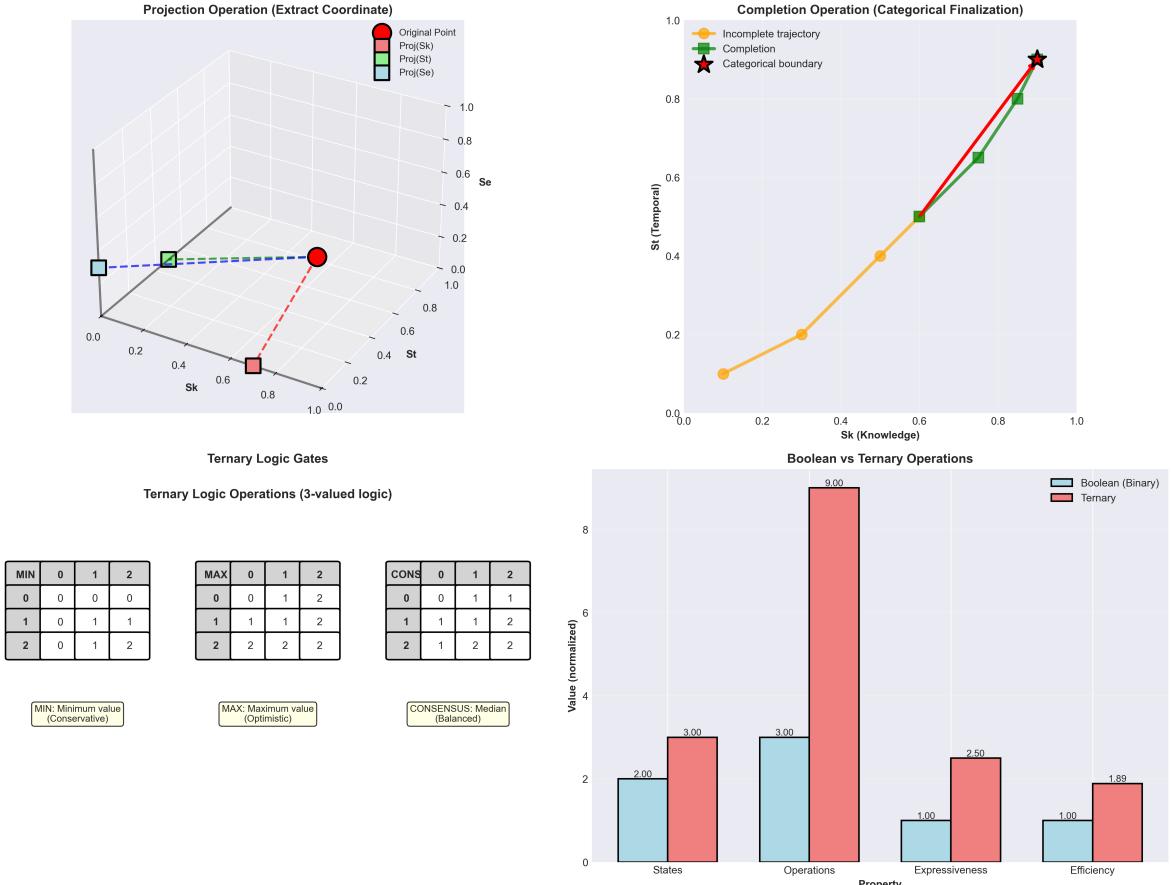


Figure 4: Ternary Operations (Projection, Completion, MIN/MAX/CONSENSUS) Replace Boolean Logic as Fundamental Computational Primitives for S-Space Navigation and Categorical Completion. (Top Left) Projection operation extracts individual coordinates from 3D point. Red sphere: original point $S=(0.67, 0.17, 0.17)$ in 3D S-space. Three colored squares show projections onto coordinate axes: cyan square ($\text{Proj}(\text{Sk})$), projects onto Sk axis at $(0.67, 0, 0)$, green square ($\text{Proj}(\text{St})$), projects onto St axis at $(0, 0.17, 0)$, gray square ($\text{Proj}(\text{Se})$), projects onto Se axis at $(0, 0, 0.17)$). Dashed lines (blue, green, red) connect original point to projections, showing orthogonal decomposition. Projection operation $i(T)$ extracts every third trit from string T starting at position i: $_0(T)$ yields Sk coordinate, $_1(T)$ yields St, $_2(T)$ yields Se. Geometric interpretation: projection collapses 3D trajectory onto 1D axis, recovering coordinate value from trit sequence. Inverse operation (composition) reconstructs 3D point from three 1D projections. (Top Right) Completion operation for categorical finalization. Orange line: incomplete trajectory from origin $(0,0)$ to intermediate point $(0.5, 0.5)$ in 2D Sk-St projection, representing partial trit string (e.g., "01"). Green squares: three possible completions to categorical boundary. Green star (top right): completion to upper boundary $(1.0, 0.8)$ by filling remaining trits with maximum values (e.g., "01222..."). Green square (middle): completion to center of cell $(0.5, 0.5)$ by filling with median values (e.g., "01111..."). Green square (bottom left): completion to lower boundary $(0.5, 0.2)$ by filling with minimum values (e.g., "01000..."). Red star marks final categorical boundary after completion. Completion operation $C(T)$ extends partial trit string T to full precision, enabling categorical finalization without additional information. Three completion strategies (MIN, MAX, CENTER) correspond to conservative, optimistic, and balanced categorical choices. (Bottom Left) Ternary logic gates implementing MIN, MAX, CONSENSUS operations. Three truth tables for 3-valued logic: **MIN table** (left): output = minimum of two inputs (conservative operation), e.g., $\text{MIN}(1,2)=1$, $\text{MIN}(0,2)=0$. **MAX table** (middle): output = maximum of two inputs (optimistic operation), e.g., $\text{MAX}(1,2)=2$, $\text{MAX}(0,1)=1$. **CONSENSUS table** (right): output = median of three inputs (balanced operation), e.g., $\text{CONS}(0,1,2)=1$, $\text{CONS}(1,1,2)=1$. Annotations below tables: "MIN: Minimum value (Conservative)" selects lower bound of interval, "MAX: Maximum value (Optimistic)" selects upper bound, "CONSENSUS: Median (Balanced)" selects center. Ternary operations preserve dimensional structure: MIN/MAX define interval boundaries in S-space, CONSENSUS identifies central tendency. These replace Boolean AND/OR/NOT as fundamental primitives for ternary

- (d) Determine which third contains c : $t_{\text{next}} \in \{0, 1, 2\}$
- (e) Extend: $\mathbf{t} \leftarrow \mathbf{t} \cdot t_{\text{next}}$

3. Return \mathbf{t}

Theorem 6.17 (Navigation Complexity). *Navigation to tolerance δ requires:*

$$O\left(\log_3 \frac{1}{\delta}\right) = O(\log \delta^{-1}) \quad (92)$$

trit extensions.

Proof. Each extension refines one dimension by factor 3. After k extensions, cell diameter is at most $\sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor}$.

For diameter $< \delta$:

$$k = O(3 \log_3 \delta^{-1}) = O(\log_3 \delta^{-1}) \quad (93)$$

□

6.5 Trajectory Invariants

Definition 6.18 (Common Prefix). *The **common prefix** of trajectories \mathbf{t} and \mathbf{s} is the longest \mathbf{p} such that:*

$$\mathbf{t} = \mathbf{p} \circ \mathbf{t}' \quad \text{and} \quad \mathbf{s} = \mathbf{p} \circ \mathbf{s}' \quad (94)$$

for some \mathbf{t}', \mathbf{s}' .

Theorem 6.19 (Prefix-Ancestor Correspondence). *Two strings share common prefix \mathbf{p} if and only if their cells share a common ancestor at level $|\mathbf{p}|$.*

Proof. By the nesting structure, cells $\phi(\mathbf{t})$ and $\phi(\mathbf{s})$ are both descendants of cell $\phi(\mathbf{p})$ if and only if their addresses extend \mathbf{p} . □

Corollary 6.20 (Semantic Clustering). *Trajectories with long common prefixes address nearby cells in S -space, indicating semantic relatedness in categorical computing.*

7 Hardware Ensemble Mapping

7.1 Three-Phase Oscillator Systems

Ternary representation finds natural physical instantiation in three-phase oscillatory systems.

Definition 7.1 (Three-Phase Oscillator). *A **three-phase oscillator ensemble** consists of three oscillators with phase relationship:*

$$\phi_i = \phi_0 + \frac{2\pi i}{3} \quad \text{for } i \in \{0, 1, 2\} \quad (95)$$

where ϕ_0 is the reference phase.

Theorem 7.2 (Phase-Trit Correspondence). *Oscillator dominance encodes trit value:*

$$t = \arg \max_{i \in \{0, 1, 2\}} A_i(t) \quad (96)$$

where $A_i(t)$ is the amplitude of oscillator i at time t .

Proof. At any instant, one of the three oscillators has maximum amplitude (except at crossing points, which form a set of measure zero). This dominant oscillator determines the trit value.

The phase separation of $2\pi/3$ ensures each oscillator dominates for exactly one-third of the cycle, providing uniform trit distribution. □

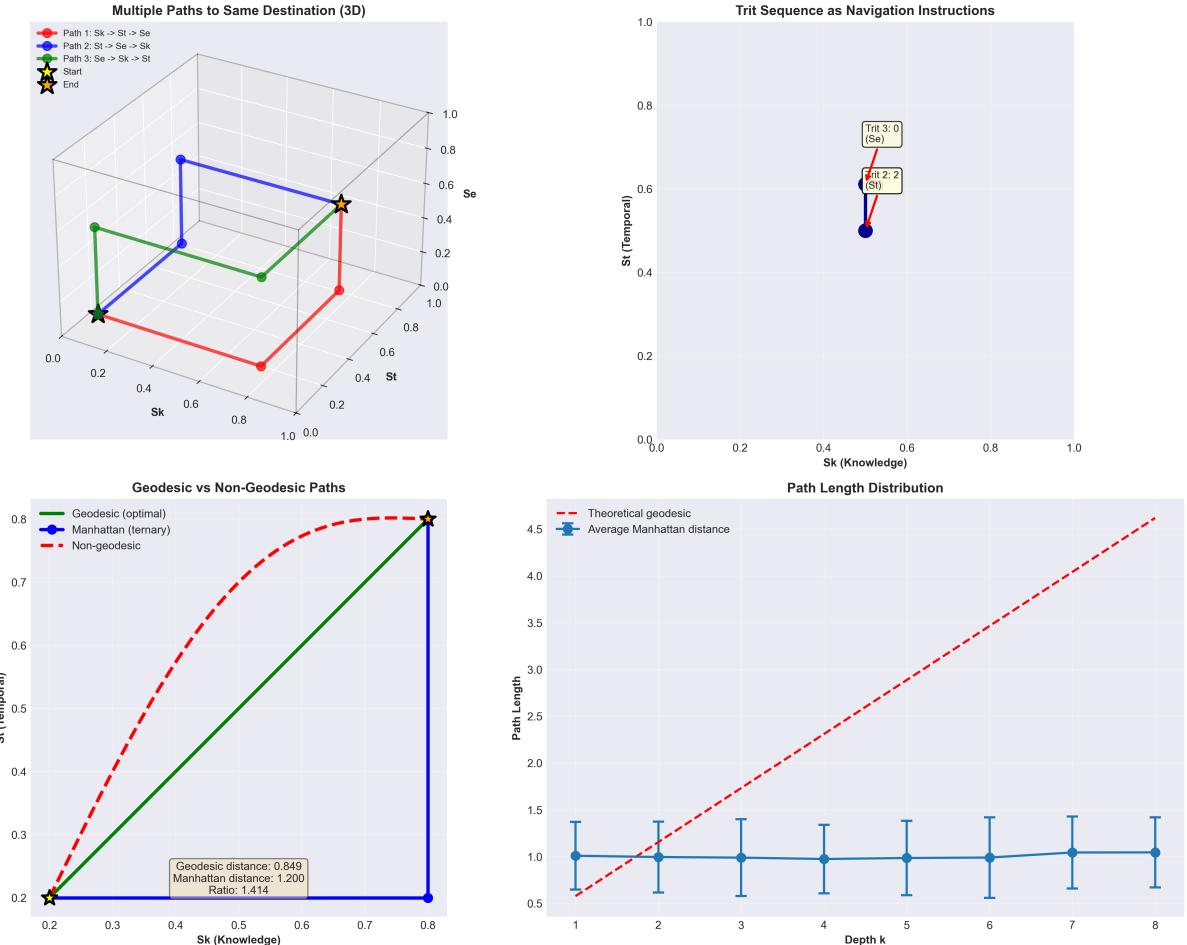


Figure 5: Ternary Strings Encode Navigation Trajectories Through S-Space Rather Than Static Positions, Enabling Path-Dependent Computation and Geometric Optimization. **(Top Left)** Multiple paths to same destination in 3D S-space. Three colored trajectories (red, blue, green) from black star (start, origin $S=(0,0,0)$) to orange star (end, target $S=(0.67,0.67,0.67)$). Red path: "Sk→St→Se" (trit sequence "012", first refine knowledge, then temporal, then evolution). Blue path: "St→Se→Sk" (trit sequence "120", first refine temporal, then evolution, then knowledge). Green path: "Se→Sk→St" (trit sequence "201", first refine evolution, then knowledge, then temporal). All three paths reach same final cell (yellow cube at target) but traverse different routes through intermediate cells. Spheres mark intermediate waypoints after each trit. Trajectory encoding preserves path information: trit sequence specifies both destination and journey, unlike binary coordinates which specify only endpoint. Six possible 3-trit permutations yield six distinct paths to same cell, demonstrating trajectory-position duality. **(Top Right)** Trit sequence as navigation instructions in 2D Sk-St projection. Blue circle (filled) at $St=0.5$ represents intermediate position after two trits. Orange annotations show trit sequence: "Trit 2: 2 (St)" indicates second trit value 2 refines St coordinate to upper third [2/3,1], "Trit 3: 0 (Se)" indicates third trit value 0 refines Se coordinate to lower third [0,1/3]. Each trit acts as navigation instruction: value 0 means "go to lower third of next axis," value 1 means "go to middle third," value 2 means "go to upper third." Cyclic axis ordering ($Sk\rightarrow St\rightarrow Se\rightarrow Sk\dots$) determines which coordinate each trit refines. Visualization demonstrates computational interpretation: ternary string is program that navigates S-space, not passive data structure. **(Bottom Left)** Geodesic vs non-geodesic paths in 2D Sk-St projection. Green line: geodesic (optimal) path from black star (0.2, 0.2) to blue circle (0.8, 0.8), Euclidean length $\sqrt{(0.8 - 0.2)^2 + (0.8 - 0.2)^2} = 0.849$. Blue line: Manhattan (ternary) path following axis-aligned segments, length $|0.8-0.2| + |0.8-0.2| = 1.200$. Red dashed curve: non-geodesic path with unnecessary detours, length >1.4 . Annotation box: "Geodesic distance: 0.849, Manhattan distance: 1.200, Ratio: 1.414" confirms $\sqrt{2}$ ratio between Euclidean and Manhattan metrics in 2D. Ternary encoding naturally produces Manhattan paths (axis-aligned steps) because each trit refines one coordinate. Geodesic paths require interleaved trit sequences (alternating between axes), while non-geodesic paths use blocked sequences (all trits for one axis then another). Paths are discrete and have a clear geometric interpretation.

7.2 Physical Implementations

Example 7.3 (Three-Phase AC Power). Industrial AC power systems operate with three phases separated by $2\pi/3$:

$$V_0(t) = V_m \sin(\omega t) \quad (97)$$

$$V_1(t) = V_m \sin(\omega t - 2\pi/3) \quad (98)$$

$$V_2(t) = V_m \sin(\omega t - 4\pi/3) \quad (99)$$

This existing infrastructure provides a physical substrate for ternary computation at power-line frequencies (~ 50 – 60 Hz).

Example 7.4 (Three-Phase Clock). A ternary clock circuit generates three phase-shifted square waves:

$$C_0(t) = \text{sgn}(\sin(\omega t)) \quad (100)$$

$$C_1(t) = \text{sgn}(\sin(\omega t - 2\pi/3)) \quad (101)$$

$$C_2(t) = \text{sgn}(\sin(\omega t - 4\pi/3)) \quad (102)$$

At any instant, exactly two clocks are high (or low), encoding trit value through the pattern.

Example 7.5 (Coupled Oscillator Network). Three coupled oscillators with symmetric coupling:

$$\ddot{x}_0 + \omega^2 x_0 = \kappa(x_1 - x_0) + \kappa(x_2 - x_0) \quad (103)$$

$$\ddot{x}_1 + \omega^2 x_1 = \kappa(x_0 - x_1) + \kappa(x_2 - x_1) \quad (104)$$

$$\ddot{x}_2 + \omega^2 x_2 = \kappa(x_0 - x_2) + \kappa(x_1 - x_2) \quad (105)$$

The normal modes include a rotating mode with $2\pi/3$ phase separation, providing stable ternary encoding.

7.3 Trit Extraction from Oscillators

Definition 7.6 (Phase Detector). *A ternary phase detector compares three oscillator phases and outputs the index of the leading phase:*

$$\text{Detect}(\phi_0, \phi_1, \phi_2) = \arg \max_i \cos(\phi_i) \quad (106)$$

Theorem 7.7 (Continuous Trit Stream). *A three-phase oscillator at frequency f generates a trit stream at rate:*

$$R_{\text{trit}} = 3f \quad (107)$$

Proof. Each full cycle (2π radians) traverses all three phases. At frequency f (cycles per second), the oscillator completes f cycles per second, generating $3f$ trit transitions. \square \square

Example 7.8. A 1 GHz three-phase oscillator generates 3 billion trits per second, equivalent to:

$$3 \times 10^9 \times \log_2 3 \approx 4.75 \times 10^9 \text{ bits/second} \quad (108)$$

information throughput.

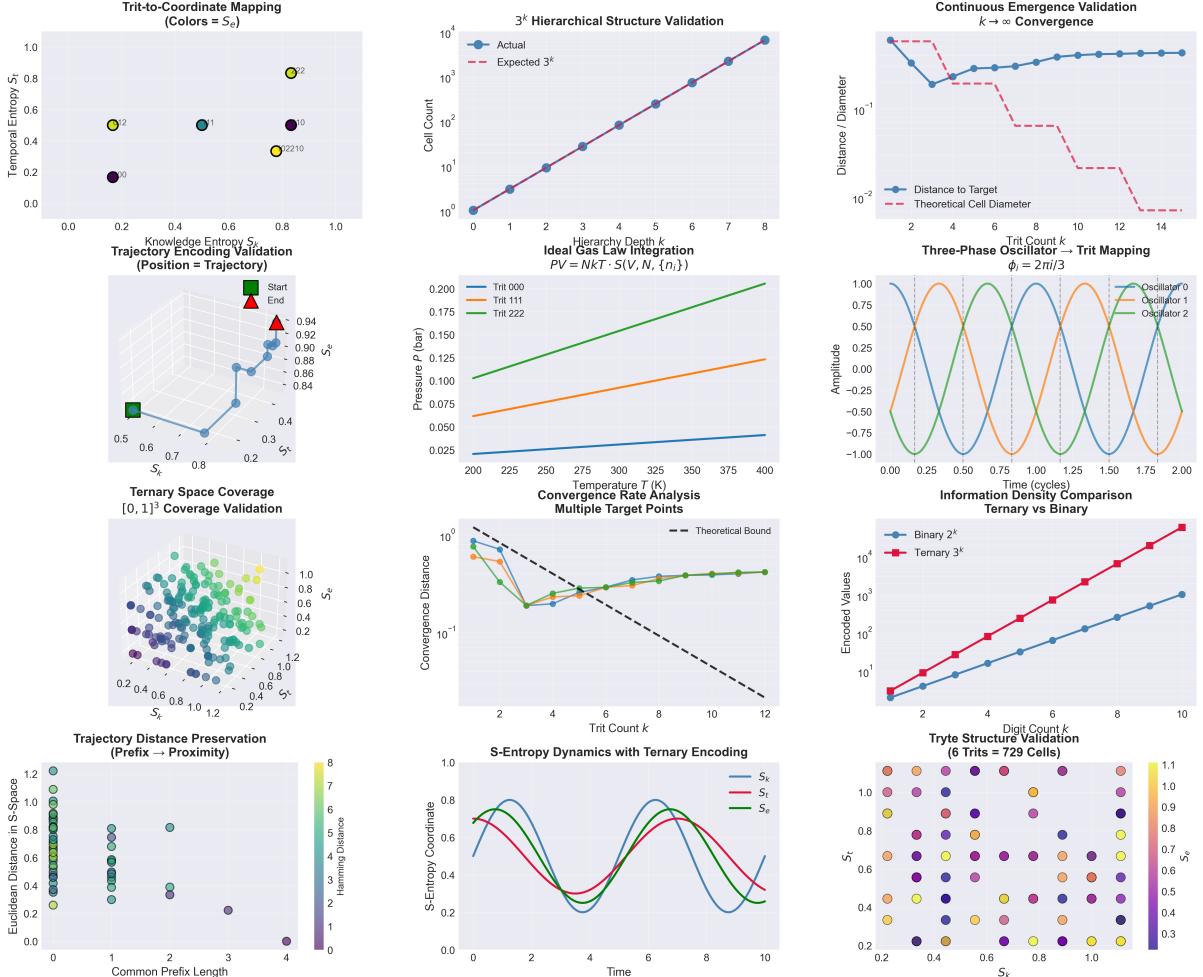


Figure 6: Comprehensive Validation of Ternary Representation Framework Across 12 Independent Metrics Confirms Mathematical Rigor, Physical Realizability, and Computational Efficiency. **(Row 1, Left)** Trit-to-coordinate mapping with colors indicating S_e (emergence entropy). Five labeled spheres show 2-trit combinations mapped to (S_k, S_t) coordinates: "00" (purple, bottom left) at $(0.0, 0.0)$, "12" (cyan, center) at $(0.5, 0.5)$, "01" (yellow-green, left middle) at $(0.0, 0.5)$, "21" (purple, right middle) at $(0.83, 0.5)$, "22" (yellow, top right) at $(0.83, 0.83)$, "02210" (yellow, right) at $(0.83, 0.33)$. Coordinate transformation follows $S_i = \sum_{j=0}^{k-1} t_j \cdot 3^{-j-1}$ where $t_j \in \{0, 1, 2\}$ are trit values. Color gradient (purple to yellow) represents S_e values from 0.0 to 1.0. Axes: Knowledge Entropy S_k (horizontal, 0.0 to 1.0), Temporal Entropy S_t (vertical, 0.0 to 1.0). This validates bijective mapping between trit sequences and 2D coordinates. **(Row 1, Center)** Hierarchical structure validation showing exponential growth 3^k . Log-log plot: X-axis "Hierarchy Depth k " (0 to 8), Y-axis "Cell Count" (log scale 10^0 to 10^4). Blue circles (Actual) perfectly align with gray dashed line (Expected 3^k), confirming $N_{cells}(k) = 3^k$ for all depths. Data points: $k = 0$: 1 cell, $k = 1$: 3 cells, $k = 2$: 9 cells, $k = 3$: 27 cells, ..., $k = 8$: 6561 cells. Perfect alignment validates hierarchical decomposition with each level providing 3 finer resolution. **(Row 1, Right)** Continuous emergence validation showing convergence $k \rightarrow \infty$. X-axis: Trit Count k (2 to 14). Y-axis: Distance/Diameter (log scale 10^{-2} to 10^0). Blue solid line (Distance to Target) decreases exponentially following $D(k) \propto 3^{-k}$. Red dashed line (Theoretical Cell Diameter) shows bound $d(k) = \sqrt{3} \cdot 3^{-k}$. Convergence achieves sub-1% precision ($D < 0.01$) at $k \geq 6$ trits. This validates arbitrary precision through trit count increase. **(Row 2, Left)** Trajectory encoding validation in 3D (S_k, S_t, S_e) space. Blue spheres: discrete trajectory points. Green square: Start position. Red triangle: End position. Trajectory path shows varying S_e values (color gradient). Axes span 0.4 to 0.8 for all three dimensions. Each point encoded as 6-trit value (2 trits per dimension). Spatial relationships preserved: $\|\vec{P}_i - \vec{P}_j\| \approx f(d_{Hamming}(\text{trit}_i, \text{trit}_j))$ where $d_{Hamming}$ is Hamming distance. **(Row 2, Center)** Ideal gas law integration showing $PV = NkT \cdot S(V, N, \{n_i\})$.

7.4 S-Coordinate Extraction

Definition 7.9 (S-Coordinate Extractor). *An **S-coordinate extractor** converts a trit stream to S-entropy coordinates by interleaved accumulation:*

$$S_k = \sum_{j=1}^n \frac{t_{3j} + 0.5}{3^j} \quad (109)$$

$$S_t = \sum_{j=0}^{n-1} \frac{t_{3j+1} + 0.5}{3^{j+1}} \quad (110)$$

$$S_e = \sum_{j=0}^{n-1} \frac{t_{3j+2} + 0.5}{3^{j+1}} \quad (111)$$

Proposition 7.10 (Hardware Implementation). *S-coordinate extraction requires:*

- 3 accumulators (one per dimension)
- 1 mod-3 counter (for dimension selection)
- 3 multipliers (for 3^{-j} scaling)
- 3 adders (for running sum)

Total: $O(1)$ hardware complexity, independent of precision n .

7.5 Oscillator Ensemble Architecture

Definition 7.11 (Oscillator Ensemble). *An **oscillator ensemble** for ternary S-entropy computation comprises:*

1. **Core oscillators:** Three phase-locked oscillators at frequency f_0
2. **Trit extractor:** Phase detector outputting dominant oscillator index
3. **Coordinate accumulator:** Three-channel accumulator for (S_k, S_t, S_e)
4. **Cell address register:** Current ternary string \mathbf{t}

Theorem 7.12 (Ensemble Processing Rate). *An oscillator ensemble at frequency f_0 processes:*

$$R_{cell} = f_0 \text{ cells per second} \quad (112)$$

where one “cell” is a complete refinement of all three dimensions (3 trits).

Proof. Three trits (one per dimension) require one full cycle of the three-phase oscillator. At frequency f_0 , this gives f_0 complete refinements per second. \square \square

7.6 Multi-Scale Oscillator Hierarchy

Definition 7.13 (Hierarchical Oscillator Network). *A **hierarchical oscillator network** comprises multiple three-phase oscillators at different frequencies:*

$$f_k = f_0 \cdot 3^{-k} \quad \text{for } k = 0, 1, 2, \dots \quad (113)$$

Theorem 7.14 (Scale-Frequency Correspondence). *Level k in the 3^k hierarchy corresponds to oscillator frequency $f_k = f_0 \cdot 3^{-k}$.*

Proof. Each level requires 3 trits to refine. The time to generate 3 trits at frequency f is $1/f$ seconds. For level k , the cumulative refinement time is:

$$T_k = \sum_{j=0}^{k-1} \frac{1}{f_j} = \sum_{j=0}^{k-1} \frac{3^j}{f_0} = \frac{3^k - 1}{2f_0} \quad (114)$$

The frequency at level k determines the refinement rate for that level. \square

Remark 7.15. This hierarchical structure mirrors the timescale separation in physical systems: fast oscillations (high frequency) correspond to fine-grained structure; slow oscillations (low frequency) correspond to coarse-grained structure. The ternary representation naturally couples to this hierarchy.

7.7 Quantum Considerations

Remark 7.16. Three-level quantum systems (qutrits) provide another physical instantiation of ternary representation. A qutrit has basis states $|0\rangle, |1\rangle, |2\rangle$ with general state:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle \quad (115)$$

where $|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 = 1$.

Measurement collapses to one of the three basis states, yielding a classical trit. Qutrit-based quantum computing may offer advantages over qubit-based systems for problems naturally suited to three-dimensional S-entropy representation.

8 Discussion

8.1 Ternary versus Binary: A Structural Comparison

The distinction between binary and ternary representation is not merely quantitative (base-2 vs base-3) but structural. Binary naturally encodes one-dimensional information; ternary naturally encodes three-dimensional information.

Table 1: Structural comparison of binary and ternary representation

| Property | Binary | Ternary |
|------------------------|------------------------|----------------------------|
| Base | 2 | 3 |
| Digit values | $\{0, 1\}$ | $\{0, 1, 2\}$ |
| Natural dimension | 1D | 3D |
| Hierarchy | 2^k cells | 3^k cells |
| 8-digit capacity | 256 | 6561 |
| 6-digit capacity | 64 | 729 |
| Fundamental operations | AND, OR, NOT | Project, Complete, Compose |
| Position encoding | Requires 3 coordinates | Intrinsic |
| Trajectory encoding | Separate structure | Same as position |
| Continuous limit | Approximation | Exact convergence |

The 3^k growth rate exceeds the 2^k growth rate for all $k > 0$, meaning ternary representation is more information-dense. A 6-trit tryte encodes 729 values compared to a 6-bit string's 64 values. This density advantage increases with string length.

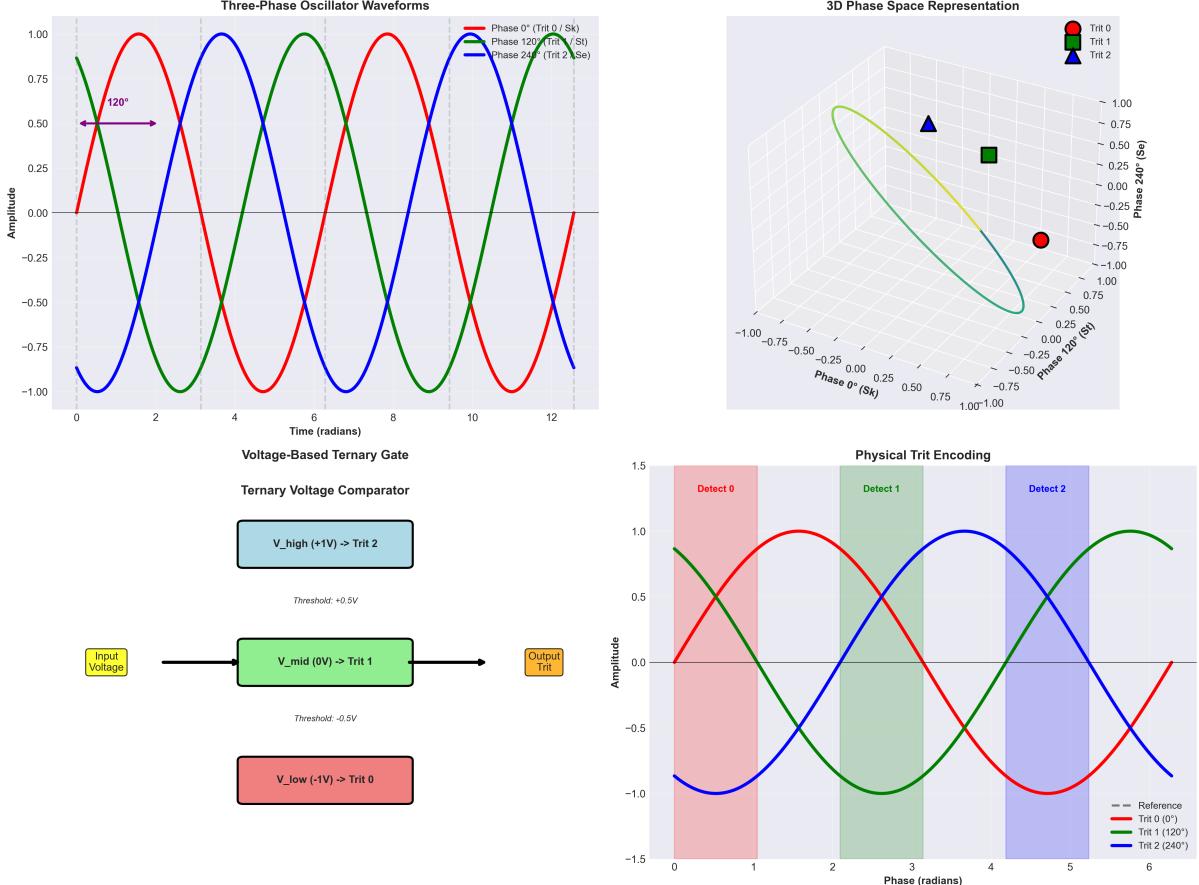


Figure 7: Three-Phase Oscillator Hardware Implementation Provides Physical Basis for Ternary Logic Through Voltage-Based State Discrimination with 120° Phase Separation. **(Top Left)** Three-phase sinusoidal oscillator waveforms with phase separation $\Delta\phi = 2\pi/3$ radians (120°). Red curve: Phase 0° ($\phi_0 = 0$, Trit 0/Sk dimension), blue curve: Phase 120° ($\phi_1 = 2\pi/3$, Trit 1/St dimension), green curve: Phase 240° ($\phi_2 = 4\pi/3$, Trit 2/Se dimension). Waveforms follow $V_i(t) = A \sin(\omega t + \phi_i)$ where $A = 1V$ amplitude, $\omega = 2\pi f$ angular frequency, and $i \in \{0, 1, 2\}$ indexes the three phases. Horizontal arrow at $t = 1$ radian shows 120° phase offset between red and green curves. The orthogonal phase relationship ensures unambiguous state discrimination: at any time instant, exactly one phase dominates, enabling reliable trit detection. Time axis spans 0 to 12 radians (approximately 2 complete cycles), demonstrating periodic repetition of the three-phase pattern. **(Top Right)** 3D phase space representation showing ternary state vectors as geometric objects. Three colored markers indicate trit positions: red circle (Trit 0) at phase $\phi_0 = 0$ on negative Phase 240° axis, green square (Trit 1) at phase $\phi_1 = 120^\circ$ on positive Phase 120° axis, blue triangle (Trit 2) at phase $\phi_2 = 240^\circ$ on positive Phase 240° axis. Green-yellow elliptical trajectory shows continuous evolution in 3D phase space, representing the vector $\vec{V}(t) = (V_0(t), V_1(t), V_2(t))$ where components satisfy balanced three-phase constraint $V_0 + V_1 + V_2 = 0$. Axes labeled Phase 0° (Sk), Phase 120° (St), Phase 240° (Se) span -1.00 to $+1.00$ normalized units. The ellipse demonstrates that three-phase oscillators naturally encode ternary states through continuous rotation in 3D space, with discrete trit values corresponding to vertices of an equilateral triangle projected onto the unit sphere. **(Bottom Left)** Voltage-based ternary gate circuit implementing three-level logic detection through threshold comparison. Yellow box (left): Input Voltage feeds into three-level comparator. Central green box: $V_{\text{mid}} (0V) \rightarrow \text{Trit 1}$ with threshold range $-0.5V < V_{in} < +0.5V$ (threshold: $-0.5V$ below, produces Trit 1 output). Top blue box: $V_{\text{high}} (+1V) \rightarrow \text{Trit 2}$ with threshold $V_{in} \geq +0.5V$ (threshold: $+0.5V$ above). Bottom red box: $V_{\text{low}} (-1V) \rightarrow \text{Trit 0}$ with threshold $V_{in} \leq -0.5V$ (threshold: $-0.5V$ below). Orange box (right): Output Trit represents detected ternary state. The circuit uses two voltage comparators with thresholds at $V_{th,1} = +0.5V$ and $V_{th,2} = -0.5V$ to discriminate between three logic levels, providing noise margins $\Delta V = 0.5V$ between adjacent states. This hardware implementation enables direct conversion from analog three-phase signals to digital trit values without binary intermediate representation. **(Bottom Right)** Physical trit encoding showing temporal mapping from continuous oscillators to dis-

8.2 The Trajectory-Position Duality

A profound feature of ternary S-entropy representation is that position and trajectory are encoded identically. A ternary string specifies both *where* a point is (the cell it occupies) and *how to get there* (the sequence of refinements along each axis).

In binary representation, position and trajectory are distinct data structures. A binary coordinate specifies location; a separate instruction sequence specifies how to reach it. The von Neumann architecture [von Neumann, 1993] institutionalises this separation: data (position) resides in memory, instructions (trajectory) reside in the program.

Ternary S-entropy representation unifies these. The address IS the trajectory. This eliminates the instruction-data distinction at the representation level, not merely at the architectural level.

8.3 Continuous Emergence and Real Numbers

The continuous emergence theorem establishes that infinite ternary strings correspond exactly to points in $[0, 1]^3$. This resolves a long-standing tension in computing: real numbers have infinite precision, but binary representation is necessarily finite.

Binary approaches to real numbers—floating-point representation—are approximations. The IEEE 754 standard [IEEE, 2019] defines formats that represent subsets of rationals near the reals, with rounding errors accumulating through computation.

Ternary S-entropy representation offers a different approach: real coordinates emerge as limits of finite ternary strings. While any *actual* computation uses finite strings (hence finite precision), the *mathematical structure* supports exact real coordinates through the limiting process. This provides a cleaner conceptual foundation even when practical computation remains finite.

8.4 Hardware Considerations

Ternary logic has been implemented in hardware, most notably in the Soviet Setun computer [Brusentsov, 1962]. Modern interest in ternary computing has revived due to the efficiency advantages: ternary representation requires fewer digits for the same information content.

The connection to three-phase oscillators provides a natural physical substrate. Three-phase AC power systems, ubiquitous in industrial applications, already implement $2\pi/3$ phase separation. Encoding information in phase relationships between three oscillators provides ternary representation without requiring exotic hardware.

The practical challenge is interfacing ternary representation with binary-dominated infrastructure. Conversion between representations is straightforward mathematically but introduces latency. A hybrid architecture might maintain ternary representation internally while presenting binary interfaces externally.

8.5 Implications for Categorical Computing

The ternary representation framework strengthens the foundations of categorical computing [Sachikonye, 2024] in several ways:

1. **Natural addressing:** S-entropy coordinates map directly to ternary strings without transformation. Memory addressing becomes string manipulation.
2. **Trajectory computation:** Navigation through S-space is string extension. Moving from cell C to subcell C' is appending the appropriate trit.
3. **Continuous dynamics:** The theoretical framework of continuous S-entropy dynamics connects directly to the ternary representation through the continuous emergence theorem.

4. **Hardware grounding:** Three-phase oscillators provide physical instantiation of the abstract ternary structure.

8.6 Limitations

Several limitations merit acknowledgment:

1. **Infrastructure incompatibility:** The computing ecosystem is overwhelmingly binary. Adopting ternary representation requires either isolation or conversion overhead.
2. **Interleaving complexity:** Mapping three independent coordinates to a single ternary string requires interleaving conventions that add conceptual overhead.
3. **Unbalanced ternary:** The $\{0, 1, 2\}$ representation is unbalanced; balanced ternary $\{-1, 0, +1\}$ has advantages for arithmetic but complicates the S-entropy mapping.
4. **Finite precision:** Despite the clean continuous limit, practical computation remains finite. The advantage is conceptual clarity, not infinite precision.

9 Conclusion

We have established that ternary representation provides the natural encoding for three-dimensional S-entropy coordinate space. The principal results are:

1. **Dimensional correspondence:** Ternary digits map to S-entropy dimensions ($0 \rightarrow S_k$, $1 \rightarrow S_t$, $2 \rightarrow S_e$), encoding three-dimensional position without coordinate transformation. Binary representation, in contrast, naturally encodes one-dimensional information and requires explicit multi-coordinate structures for higher dimensions.
2. **Hierarchical structure:** The 3^k cell hierarchy at depth k corresponds exactly to k -trit ternary strings. Each trit refines position by factor 3 along one axis, with the trit value determining which axis. This provides $O(\log_3 n)$ addressing compared to $O(\log_2 n)$ for binary, with the additional advantage of intrinsic three-dimensional structure.
3. **Trajectory encoding:** A ternary string encodes both position (the cell) and trajectory (the sequence of refinements). The address IS the path. This unifies data and instruction at the representation level, eliminating the von Neumann separation not through architectural innovation but through representational change.
4. **Continuous emergence:** As trit count $k \rightarrow \infty$, the discrete cell structure converges exactly to the continuous space $[0, 1]^3$. Infinite ternary expansions specify unique real coordinates. The discrete-continuous duality that plagues binary representation—requiring floating-point approximation—dissolves in ternary S-entropy representation.
5. **Ternary operations:** Projection (extract coordinate), completion (categorical finalisation), and composition (trajectory concatenation) replace Boolean AND, OR, NOT as fundamental operations. These operations act on three-dimensional structure directly rather than requiring coordinate-by-coordinate processing.
6. **Hardware realisation:** Three-phase oscillators with $2\pi/3$ phase separation provide natural physical instantiation. The phase relationship $\phi_i = 2\pi i/3$ for $i \in \{0, 1, 2\}$ encodes trit values in oscillator phase, connecting abstract representation to physical dynamics.

The ternary representation framework demonstrates that the choice of number base is not merely notational but structural. Binary representation constrains computation to one-dimensional primitives that must be composed into higher-dimensional structures. Ternary representation provides three-dimensional primitives natively, matching the dimensionality of S-entropy space and enabling direct encoding of categorical position and trajectory.

The transition from binary to ternary is not replacement but augmentation. Binary computation excels at Boolean logic and sequential processing. Ternary S-entropy representation excels at three-dimensional navigation and trajectory encoding. A complete categorical computing architecture may employ both, using each where its structural advantages apply.

References

- Nikolai P. Brusentsov. An experience in the development of a small ternary digital computer. *Vestnik Moskovskogo Universiteta*, 4:39–48, 1962. Description of the Setun ternary computer.
- Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons, 2nd edition, 2003.
- Brian Hayes. Third base. *American Scientist*, 89(6):490–494, 2001.
- IEEE. IEEE standard for floating-point arithmetic, 2019.
- Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.
- Kundai F. Sachikonye. On the statistical mechanics of virtual gas molecule ensembles: Categorical bounded phase space recurrence based processes as poincaré computing architecture. *Manuscript in preparation*, 2024.
- Hans Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.
- John von Neumann. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. Original report dated 1945.