Ternary Unit Representation: From Discrete Trits to Continuous S-Entropy Coordinates

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

December 27, 2025

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 $k \to \infty$, 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

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 (**?**) comprises three fundamental dimensions:

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

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

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

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:

$$\mathtt{t} = 0 \leftrightarrow \text{refinement along } S_k \tag{4}$$

$$\mathtt{t} = 1 \leftrightarrow \text{refinement along } S_t \tag{5}$$

$$\mathtt{t} = 2 \leftrightarrow \text{refinement along } S_e \tag{6}$$

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 $k \to \infty$, these cells become arbitrarily fine, and the ternary expansion converges to a unique point in the continuous space $[0,1]^3$:

$$\lim_{k\to\infty} \text{Cell}(\mathtt{t}_1, \mathtt{t}_2, \ldots, \mathtt{t}_k) = \mathbf{S} \in [0,1]^3 \tag{7}$$

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

1.6 Notation

Throughout this paper:

- $\mathsf{t}_i \in \{0,1,2\}$: a ternary digit (trit) at position i
- $\mathsf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_6)$: a ternary byte (tryte) of 6 trits
- $\mathcal{S} = [0,1]^3$: the S-entropy coordinate space
- $\mathbf{S} = (S_k, S_t, S_e)$: a point in S-space
- $\mathcal{C}_k$: the set of 3^k cells at depth k
- $\phi : \{0,1,2\}^k \to \mathcal{C}_k$: 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:*

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

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. □ □

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 $\gamma : [0,1] \to \mathcal{S}$ with $\gamma(0) = \mathbf{S}_1$ and $\gamma(1) = \mathbf{S}_2$.*

Proof. The straight-line path:

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

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

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:*

$$\mathcal{C}_k = \{C_{i_1,i_2,\ldots,i_k} : i_j \in \{0,1,2\}\} \tag{10}$$

where each cell $C_{i_1,\ldots,i_k}$ 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:*

$$k \equiv 0 \pmod 3 : \textit{refine } S_k \textit{ dimension} \tag{11}$$

$$k \equiv 1 \pmod 3 : \textit{refine } S_t \textit{ dimension} \tag{12}$$

$$k \equiv 2 \pmod 3 : \textit{refine } S_e \textit{ dimension} \tag{13}$$

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 $k = 0$:

$$|\mathcal{C}_k| = 3^k \tag{14}$$

□ □

2.4 Cell Addressing

Definition 2.8 (Cell Address). *A cell $C \in \mathcal{C}_k$ is addressed by a string $(i_1, i_2, \ldots, i_k)$ 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 $j \to j+1$: Each cell splits into 3 subcells, indexed by $i_j \in \{0,1,2\}$

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

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 $C \in \mathcal{C}_k$:*

$$C' \subseteq C \iff \textit{address}(C') \textit{ extends } \textit{address}(C) \tag{15}$$

Proof. If address$(C) = (i_1, \ldots, i_k)$ and address$(C') = (i_1, \ldots, i_k, i_{k+1}, \ldots, i_{k'})$, 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 address(C') extends address(C). □ □

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 $(i_1, i_2, \ldots, i_k)$, the cell centre has coordinates:*

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

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

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

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. □ □

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 $k \to \infty$.

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 $\mathbf{t} = (t_1, t_2, \ldots, t_k)$ or simply $t_1 t_2 \cdots t_k$.*

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

3.2 Ternary Numerical Representation

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

$$x = \sum_{i=1}^{\infty} t_i \cdot 3^{-i} = 0.t_1 t_2 t_3 \ldots_3 \tag{19}$$

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:

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

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

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 $\overline{0}$ and one ending in $\overline{2}$).*

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:

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

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

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

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 $m/3^n$. □ □

3.3 Information Content

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

$$I_k = k \log_2 3 \approx 1.585k \text{ bits} \tag{23}$$

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

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

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:

$$k_2 = \log_2 N \text{ bits} \tag{25}$$

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

Ternary requires only 63.1% as many digits. □ □

Remark 3.9. This efficiency advantage, noted by **?**, 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.

3.4 The Tryte: Ternary Byte

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

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

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 $\to$ resolution $1/9$)*
- *Trits 2, 5: refine S_t (2 refinements $\to$ resolution $1/9$)*
- *Trits 3, 6: refine S_e (2 refinements $\to$ resolution $1/9$)*

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

$$(\mathsf{T}_1, \mathsf{T}_2, \ldots, \mathsf{T}_n) \in (\mathbb{T}^6)^n \cong \mathbb{T}^{6n} \tag{28}$$

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 $\{\bar{1}, 0, 1\}$):*

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

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 $\mathbf{b} = (b_1, \ldots, b_n) \in \{0, 1\}^n$ 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:*

$$v = 3q_1 + t_k \tag{30}$$

$$q_1 = 3q_2 + t_{k-1} \tag{31}$$

$$\vdots \tag{32}$$

$$q_{k-1} = 3 \cdot 0 + t_1 \tag{33}$$

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:*

$$\phi_k : \mathbb{T}^k \to \mathcal{C}_k \tag{34}$$

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 $\phi_k : \mathbb{T}^k \to \mathcal{C}_k$ 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 $d = k \mod 3$:

$$C = C_0 \cup C_1 \cup C_2 \tag{35}$$

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

Define:

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

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

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:

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

□ □

4.2 The Dimension Correspondence

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

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

(with the convention $0 \mod 3 = 0$ 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* $\mathbf{t} = (t_1, t_2, \ldots, t_k)$*, the cell centre coordinates are:*

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

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

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

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:

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

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

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

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

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

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

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:

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

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

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

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:*

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

where $d_j = j \mod 3$ *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 $d_j = j \mod 3$:

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

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

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

4.5 The Metric Correspondence

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

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

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. □ □

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* $\phi : \mathbb{T}^\infty \to [0,1]^3$ *defined by:*

$$\phi(t_1, t_2, t_3, \ldots) = \lim_{k \to \infty} \phi_k(t_1, \ldots, t_k) \tag{53}$$

is well-defined and surjective.

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

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

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

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

$$\bigcap_{k=0}^{\infty} C_k = \{\mathbf{S}\} \tag{55}$$

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

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

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

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

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

For any open ball $B_\epsilon(\mathbf{S}) \subset [0,1]^3$, we need $\phi^{-1}(B_\epsilon(\mathbf{S}))$ to be open in $\mathbb{T}^\infty$.
Choose k such that $\sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor} < \epsilon$. Let $\mathbf{t} \in \phi^{-1}(B_\epsilon(\mathbf{S}))$ with $\phi(\mathbf{t}) = \mathbf{S}' \in B_\epsilon(\mathbf{S})$.
The basis set $U_{t_1,\ldots,t_k}$ contains $\mathbf{t}$. For any $\mathbf{t}' \in U_{t_1,\ldots,t_k}$:

$$d(\phi(\mathbf{t}'), \phi(\mathbf{t})) \leq \sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor} < \epsilon \tag{57}$$

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

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:*

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

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

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

trits.

Proof. A k-trit string addresses a cell of diameter at most $\sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor}$.
For $d(\phi(\mathbf{t}), \mathbf{S}) < \delta$, we need:

$$\sqrt{3} \cdot 3^{-\lfloor k/3 \rfloor} < \delta \tag{60}$$

Solving:

$$\lfloor k/3 \rfloor > \log_3 \frac{\sqrt{3}}{\delta} \implies k \geq 3 \left\lceil \log_3 \frac{\sqrt{3}}{\delta} \right\rceil \tag{61}$$

□ □

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

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

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

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* $\{0,2\}$*:*

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

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* $\phi : \mathbb{T}^\infty \to [0,1]^3$ *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 **??**. □ □

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 (**??**) is deep. While Peano's original curve uses a continuous surjection $[0,1] \to [0,1]^2$, our construction uses the product structure of $\mathbb{T}^\infty$ 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* $\mathbb{T}^\infty$ *(with uniform distribution on each* $\{0,1,2\}$*).*

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

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

with product measure 3^{-k}.

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

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

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

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 (**?**). □ □

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.

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:*

$$t = 0 : \textit{step along } S_k \textit{ (refine knowledge)} \tag{66}$$

$$t = 1 : \textit{step along } S_t \textit{ (refine temporal)} \tag{67}$$

$$t = 2 : \textit{step along } S_e \textit{ (refine evolution)} \tag{68}$$

Definition 6.2 (Trajectory). *A **trajectory** is a sequence of trits $\mathbf{t} = (t_1, t_2, \ldots, t_k)$ 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 $(t_1, t_2, \ldots, t_k)$ specifies the unique path:

$$\mathcal{S} = C_0 \supset C_1 \supset \cdots \supset C_k = \phi(\mathbf{t}) \tag{69}$$

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

The address IS the path. No additional structure is needed. □ □

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:*

$$\pi_d(\mathbf{t}) = (t_j : j \equiv d \pmod 3) \tag{70}$$

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

$$\pi_0(\mathbf{t}) = (t_3, t_6) = (2, 0) \quad (S_k \text{ refinements}) \tag{71}$$

$$\pi_1(\mathbf{t}) = (t_1, t_4) = (1, 2) \quad (S_t \text{ refinements}) \tag{72}$$

$$\pi_2(\mathbf{t}) = (t_2, t_5) = (0, 1) \quad (S_e \text{ refinements}) \tag{73}$$

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

$$\mathbf{t} \cdot t = (t_1, \ldots, t_k, t) \tag{74}$$

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

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

$$\mathbf{t}|_j = (t_1, \ldots, t_j) \tag{75}$$

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

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

$$\mathbf{t} \circ \mathbf{s} = (t_1, \ldots, t_k, s_1, \ldots, s_m) \tag{76}$$

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

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

$$(\mathbf{t} \circ \mathbf{s}) \circ \mathbf{r} = \mathbf{t} \circ (\mathbf{s} \circ \mathbf{r}) \tag{77}$$

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:*

$$\Omega(\mathbf{t}) = (t_1, t_2, \ldots, t_k, c_{k+1}, c_{k+2}, \ldots) \tag{78}$$

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

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

$$\Omega_{\text{centroid}}(\mathbf{t}) = (t_1, \ldots, t_k, 1, 1, 1, \ldots) \tag{79}$$

This completes to the centre of the current cell.

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

$$\Omega_{\min}(\mathbf{t}) = (t_1, \ldots, t_k, 0, 0, 0, \ldots) \tag{80}$$

This completes to the minimal-coordinate corner.

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

$$\Omega_{\max}(\mathbf{t}) = (t_1, \ldots, t_k, 2, 2, 2, \ldots) \tag{81}$$

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: $d = (|\mathbf{t}| + 1) \mod 3$
   (b) Extract target coordinate: $c = \mathbf{S}_{\text{target}}[d]$
   (c) Compute current interval bounds for dimension d
   (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 t**

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

$$O\left(\log_3 \frac{1}{\delta}\right) = O(\log \delta^{-1}) \tag{82}$$

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 $< \delta$:

$$k = O(3 \log_3 \delta^{-1}) = O(\log_3 \delta^{-1}) \tag{83}$$

□ □

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:*

$$\mathbf{t} = \mathbf{p} \circ \mathbf{t}' \quad \textit{and} \quad \mathbf{s} = \mathbf{p} \circ \mathbf{s}' \tag{84}$$

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:*

$$\phi_i = \phi_0 + \frac{2\pi i}{3} \quad \textit{for } i \in \{0, 1, 2\} \tag{85}$$

where ϕ_0 is the reference phase.

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

$$t = \arg \max_{i \in \{0,1,2\}} A_i(t) \tag{86}$$

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. □ □

7.2 Physical Implementations

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

$$V_0(t) = V_m \sin(\omega t) \tag{87}$$
$$V_1(t) = V_m \sin(\omega t - 2\pi/3) \tag{88}$$
$$V_2(t) = V_m \sin(\omega t - 4\pi/3) \tag{89}$$

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:

$$C_0(t) = \text{sgn}(\sin(\omega t)) \tag{90}$$
$$C_1(t) = \text{sgn}(\sin(\omega t - 2\pi/3)) \tag{91}$$
$$C_2(t) = \text{sgn}(\sin(\omega t - 4\pi/3)) \tag{92}$$

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:

$$\ddot{x}_0 + \omega^2 x_0 = \kappa(x_1 - x_0) + \kappa(x_2 - x_0) \tag{93}$$
$$\ddot{x}_1 + \omega^2 x_1 = \kappa(x_0 - x_1) + \kappa(x_2 - x_1) \tag{94}$$
$$\ddot{x}_2 + \omega^2 x_2 = \kappa(x_0 - x_2) + \kappa(x_1 - x_2) \tag{95}$$

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:*

$$\mathit{Detect}(\phi_0, \phi_1, \phi_2) = \arg\max_i \cos(\phi_i) \tag{96}$$

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

$$R_{trit} = 3f \tag{97}$$

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. □ □

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

$$3 \times 10^9 \times \log_2 3 \approx 4.75 \times 10^9 \text{ bits/second} \tag{98}$$

information throughput.

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:*

$$S_k = \sum_{j=1}^{n} \frac{t_{3j} + 0.5}{3^j} \tag{99}$$

$$S_t = \sum_{j=0}^{n-1} \frac{t_{3j+1} + 0.5}{3^{j+1}} \tag{100}$$

$$S_e = \sum_{j=0}^{n-1} \frac{t_{3j+2} + 0.5}{3^{j+1}} \tag{101}$$

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:*

$$R_{cell} = f_0 \text{ cells per second} \tag{102}$$

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. □ □

7.6 Multi-Scale Oscillator Hierarchy

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

$$f_k = f_0 \cdot 3^{-k} \quad \text{for } k = 0, 1, 2, \ldots \tag{103}$$

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:

$$T_k = \sum_{j=0}^{k-1} \frac{1}{f_j} = \sum_{j=0}^{k-1} \frac{3^j}{f_0} = \frac{3^k - 1}{2f_0} \tag{104}$$

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

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:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle \tag{105}$$

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.

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 (**?**) 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 (**?**) 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 (**?**). 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 (**?**) 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 ($0 \to S_k$, $1 \to S_t$, $2 \to S_e$), 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 $k \to \infty$, 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.