EMx as a Universal Tokenizer: A Discrete-Continuous Event Algebra for Text, Image, and Symbolic Streams

Shawn Hohol 11-14-2025

Abstract

This paper presents a constructive demonstration of how the EMx operator framework can function as a universal tokenizer across domains such as text, images, audio, and symbolic sequences. EMx does not rely on subword segmentation or byte-pair encoding. Instead, tokens emerge as structured EMx events defined by lattice coordinates, operator gates, phase information, and dynamic metrics. The system's 4—2—4 packet structure allows raw input to be folded into a unified discrete-continuous format capable of representing semantic roles, structural forces, closure properties, symmetry constraints, and NULL-reservoir behavior. We outline the EMx tokenization pipeline, the packet dynamics, and worked examples for both English text and a Voynich star label. The result shows that EMx is not merely capable of tokenization: tokenization is a natural consequence of its operator algebra.

1 Introduction

Traditional natural language tokenizers segment text by string patterns such as byte-pair encoding (BPE) or unigram models. These systems ignore geometry, phase, operator structure, and dynamic state evolution. EMx provides a radically different approach: tokens are discrete events expressed in a structured operator algebra.

An EMx token has the form:

$$\text{Token} = (\text{coords}, \text{operator}, \text{phase}, \alpha, \beta, \gamma, \Omega, \varnothing)$$

where:

- coordinates lie on a ternary lattice T_0–T_4,
- operators encode dynamical roles (Lift, Exchange, Collapse, Normalize),
- phase follows EMx's 7-phase loop and 96-tick orbit,
- metrics $(\alpha, \beta, \gamma, \Omega, \varnothing)$ measure structure, curvature, closure, no-clone status, and NULL-reservoir load.

This makes EMx inherently multi-modal and domain-independent. Any input converted to seed coordinates can be tokenized by EMx.

2 The EMx Token Concept

In EMx, a token is not a subword but a structured event:

$$\text{event} \equiv (\text{coords on } T_0\text{–}T_4,\ H,\ \text{phase},\ \alpha, \beta, \gamma, \Omega, \varnothing)$$

Coordinates encode location in the ternary geometry. H is an operator gate drawn from:

- Lift,
- Exchange,
- Collapse,
- Normalize.

Phase determines where on EMx's 96-tick cycle the event is sampled. The metrics reflect dynamic evolution:

- α: structural weight,
- β: curvature or drift,
- γ: closure consistency,
- Ω: no-clone safeguard,
- $\varnothing$: NULL-reservoir accumulation.

These are the core variables of EMx tokenization.

3 4—2—4 Packet Architecture

EMx uses a 10-bit structured packet:

$$W_3 W_2 W_1 W_0 \ \big|\ H_1 H_0 \ \big|\ E_3 E_2 E_1 E_0$$

- W encodes "what/where" information, mapping to direction or class on the T_3/T_4 lattice.
- H selects the operator gate (Lift, Exchange, Collapse, Normalize).
- E carries echo/parity to implement Ω and $\varnothing$ constraints.

The packet interacts with the EMx field during the "carrier window" (||), which corresponds to the moment when packet bits update under operator action. This is the moment of tokenization itself.

4 General EMx Tokenization Pipeline

EMx can tokenize any domain by a simple adapter:

raw input → features → seed coords → 4—2—4 packets → EMx orbit → token stream.

4.1 Step 1: Feature Extraction

Domain-specific but minimal:

- text: characters, graphemes, word segments,
- image: edges, patches, color clusters,
- audio: short-time spectra, onset events.

Each feature obtains a symbolic ID.

4.2 Step 2: Mapping Features to T_0 Coordinates

A static mapping assigns each feature a ternary coordinate $(x, y, z) \in \{-1, 0, 1\}^3$. This determines initial EMx state placement.

4.3 Step 3: Packet Construction

For each feature:

- W encodes direction/class from (x, y, z),
- H is chosen from contextual role (Lift, Exchange, Collapse, Normalize),
- $E = \text{Gray}(W)$ or $\text{mirror}(W)$, providing parity and stability.

4.4 Step 4: Running Packets Through the EMx Orbit

Packets evolve through the 7-phase loop and 96 ticks. W changes under H, and Ω prevents cloning. $\varnothing$ absorbs residuals.

4.5 Step 5: Sampling at T_2

At fixed readout phases, collapse to discrete token IDs:

$$\text{token\_id} = \text{hash}(W_{\text{final}}, H, \text{phase}),$$

along with the metrics $(\alpha, \beta, \gamma, \Omega, \varnothing)$.

4.6 Step 6: Final Token Stream

Resulting tokens are fully structured EMx events:

$$[(\mathrm{id}_1, \alpha_1, \beta_1, \gamma_1, \Omega_1, \varnothing_1), \dots].$$

5 Example A: English Phrase "herb fire"

5.1 Seed Coordinates

Assume:

$$\text{"herb"} \mapsto (-0, +0, 0), \quad \text{"fire"} \mapsto (+0, +0, 0).$$

5.2 Packet Construction

Word-initial "herb":

$$W = 0101, \quad H = 00 \text{ (Lift)}, \quad E = 0111.$$

Modifier "fire":

$$W = 1010, \quad H = 01 \text{ (Exchange)}, \quad E = 1111.$$

5.3 EMx Orbit Effects

"herb" stabilizes under Lift, producing low β and high γ. "fire" drifts under Exchange, producing higher β.

Tokens become:

$$\mathrm{token}_1 = (\mathrm{id}_1, \alpha \approx 0.33, \beta \approx 0.18, \gamma \approx 0.999),$$

$$\mathrm{token}_2 = (\mathrm{id}_2, \alpha \approx 0.66, \beta \approx 0.42, \gamma \approx 0.996).$$

6 Example B: Voynich Star Label "doaro"

"doaro" is treated as a symbolic unit with astronomical context.

6.1 Seed Mapping

$$\text{"doaro"} \mapsto (0, +0, +0)$$

with context tagged as dynamic (Exchange operator).

6.2 Packet Construction

$$W = 0110, \quad H = 01 \text{ (Exchange)}, \quad E = 0110.$$

6.3 EMx Orbit Effects

The packet moves along the T_4 shell, sampling at T2 yields:

$$\text{token} = (\text{id}, \alpha, \beta, \gamma, \Omega, \varnothing).$$

Thus a Voynich glyph sequence becomes comparable to an English word using the same token system.

7 EMx Compared to Classical Tokenizers

EMx differs fundamentally from BPE:

- BPE is purely symbolic; EMx combines geometry, phase, and operator action.
- EMx embeds semantics through α, β, γ.
- EMx embeds structure through lattice positions.
- EMx embeds dynamics through the 96-tick orbit.
- EMx embeds invariants through Ω and $\varnothing$.

EMx effectively replaces:

- character-level tokenizers,
- subword-level segmenters,
- morphological analyzers,
- semantic encoders.

8 Conclusion

EMx is a true universal tokenizer:

- any domain becomes tokens by a simple feature $\rightarrow$ ternary mapping;
- EMx packets evolve through a geometric-dynamical operator system;
- tokens carry semantic, structural, and dynamical signatures;
- EMx unifies text, audio, image, and symbolic domains.

The framework does not approximate tokenization—it produces tokens as an emergent property of its operator algebra. EMx thus forms a structured computational substrate suitable for symbolic models, neural models, and hybrid architectures.