The Market Is Talking:
A Market-Driven Generative Language Engine

FelixFelixFelix
hello@felixfelixfelix.com

Abstract. We describe a deterministic generative artwork in which two curated baskets of already-deployed ERC-20 markets behave as the two hidden layers of a *deep* recurrent reservoir, and a small set of further ERC-20 markets act as its input neurons. The network's wiring is not authored but derived from market structure (price and activity correlation *within* each layer and *cross-correlation* between them) and drifts as the market drifts. One discrete update per block evolves the two hidden states; a market-derived readout projects their concatenation onto a vocabulary; and a five-layer deterministic grammar assembles the winning words into an accreting line of language whose syntax is well-formed by construction and whose meaning is a controlled distance from coherence: tight when the market is calm, surreal when it is violent. Every choice, which production to expand, which word fills a slot, where a line breaks, is resolved from a fixed-point function of network state, with no floating point and no random source. Consequently any machine that re-derives the same block window reproduces the same text mark for mark. This paper documents the system as built: the data read from chain, the reaction and two-layer reservoir that turn it into network state, the vocabulary, the grammar, and the generative sentence algorithm.

1. Introduction

The premise is taken literally: the market is not the subject of the artwork, the market *is* the artwork, and the artwork *speaks*. Four existing ERC-20 markets are the input neurons, the voice that can pushed by their trading. Two curated baskets of existing ERC-20 markets, ones the artist does not control, are the hidden or reservoir neurons: the involuntary, ambient substrate. The synaptic weights between them are not hand-authored; they are derived from market structure and drift as correlations drift [4]. The network is a recurrent reservoir [1, 3, 4, 7], so it has an intrinsic rhythm of its own; live trading perturbs and corrupts that rhythm and outputs words.

Two design commitments shape everything below.

1.1 A*gency versus substrate*

The artist and collectors can only move the few input editions, while the vast hidden body is the indifferent market. Steering the piece means trading; the rest is ambient noise. As a direct consequence, a roaring ambient market drowns an individual's steering, while a calm market lets a single input’s theme dominate.

1.2 Determinism as a contract

The state at block N is a seed-derived initial condition iterated block by block from a pinned start, with every block's inputs reconstructed from sealed chain history. The browser executes a deterministic trajectory rather than producing one. Because a chaotic recurrent map amplifies last-bit differences and floating-point results are not bit-reproducible across platforms [21], the entire pipeline runs in scaled-integer (*BigInt*) fixed-point math with a deterministic integer *tanh*; no wall-clock, no unseeded randomness, no

float reductions whose order varies. A live stream and a minted recording of the same window therefore agree bit for bit.

The renderer is a chain of vanilla-JavaScript modules under the *MIT* namespace, gathered for publication into two files, *engine.js* (the sentence engine) and *renderer.js* (the visual and audio surface), with the operator-set endpoints served from the main smart-contract on load and during run time via event listeners.

The engine is driven once per block, as in Figure 1.

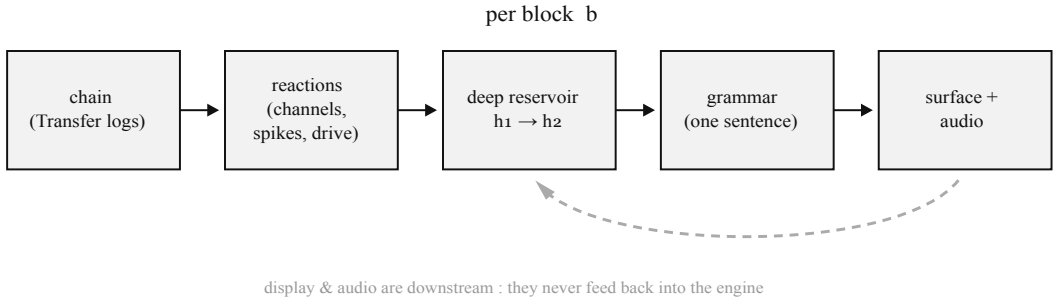

Figure 1. The per-block pipeline. Each new block drives the reactions, the two-layer reservoir, and the grammar in turn; the surface and audio layers are strictly downstream and never feed back into the engine, so they may use floating-point math while the engine does not.

The reservoir and grammar never learn *how* the data is wired or where it comes from, so a given block produces the same words however its logs are re-derived. The display and audio layers render a smooth sub-block reveal but are read-only consumers of state. Sections 2 and 3 produce the network state, Section 4 turns it into language.

2. The Composition: Data Read from the Chain

The chain is the hub. Everything dynamic is reconstructed in the browser from on-chain logs; nothing is stored between reads. The same routine reconstructs state for the live view and for the sealed recording, which is what guarantees that a given block yields byte-identical observations everywhere.

2.1. One source, one reconstruction path

MIT.sources exposes a single interface, *run({ onBlock, onStatus })*. The live source uses a websocket *newHeads* subscription as the block clock; for each head it fetches that block's logs over HTTP and steps the engine, backfilling a look-back window first so the reservoir memory is warm. All fetching flows through one routine, *rangeObs()*, which means the live stream and any later re-derivation of the same range produce identical observations: deterministic by construction. The HTTP transport spaces call-starts and retries with exponential backoff over a failover list of endpoints, so the stream rides through rate-limits without diverging.

2.2. The datum and the observation record

For every basket token and every input token, the source pulls *Transfer(from, to, value)* logs for the block range and groups them by block. Each log is classified by endpoint: a mint (*from = 0x0*) gives direction *+1* and *+value* net flow; a burn (*to = 0x0*) gives *−1* and *−value*; a plain transfer gives direction *0* and contributes to volume and count only.

Every block in the range is emitted even if it carried no events. The clock ticks regardless, so the network keeps evolving on its intrinsic rhythm during market silence. Each entity's logs in a block reduce to one observation record, the only structure the reaction layer consumes:

Table 1. Per-entity, per-block observation record.

field	meaning	source
volume	Σ |trade size|	Transfer values
netFlow	Σ signed size (mint +, burn −)	Transfer mint / burn
count	number of events	Transfer count
curveProgress	S / Smax (inputs only)	edition state
swaps[]	per-event { size, dir } impulses	individual logs

For each token of the input and reservoirs, the signal is driven entirely by ERC-20 transfer *activity*: volume, count, mint/burn flow, and per-transfer impulses.

2.3. The composition, and the three roles of an event

The work wires a concrete network. The first hidden layer is a basket of $M_1 = 10$ broad-market tokens, the ambient context (WETH, WBTC, DAI, USDC, LINK, UNI, AAVE, CRV, LDO, COMP). The second hidden layer is a basket of $M_2 = 10$ more volatile tokens, the surface that actually speaks (PEPE, SHIB, SNX, SUSHI, 1INCH, YFI, BAL, CVX, FXS, RPL). The input layer is $K = 4$ tokens carrying the themes ENTROPY, SOVEREIGN, NETWORK and SILENCE, and the vocabulary is $V = 299$ words. Each token both *drives* its node and helps *wire* its layer's recurrent matrix. The same logs serve three purposes at once: *deterministic reconstruction* (re-derived in order, the logs rebuild exact state at every block), *discrete impulses* (each event is injected as its own spike, not a smoothed average, giving the corruption its irregular texture), and *per-block drive* (the steady aggregate the spikes ride on).

3. From Market Data to Network State

This section bridges the chain and the language engine: it turns the observation record into the quantities the grammar consumes: the hidden state, the word logits, the activity scalar A, the volatility v, and the spike magnitude. All of it is fixed-point.

3.1. Fixed-point arithmetic

A real number r is stored as the integer $round(r \times SCALE)$ with $SCALE = 1e9$; multiplication truncates toward zero and division is $(a \cdot SCALE)/b$. Floats appear only in two one-shot, machine-identical places: converting authored constants on initialization and rendering for display. Two primitives carry the dynamics: an integer *tanh*, a Padé-style rational approximation $\tanh(x) \approx x(27 + x^2)/(27 + 9x^2)$ for $|x| < 3$, clamped to ± 1 outside (the squashing nonlinearity φ used everywhere); and an integer Newton *sqrt* for norms and standard deviations.

3.2. Reactions: react to surprise, not size

Raw observables live on wildly different scales, so each is normalized against the token's *own* recent behaviour (an EWMA at rate $\alpha = 0.05$) and squashed by φ into $(-1, 1)$. A dormant token that suddenly

trades produces a large reaction; a constantly-churning token's individual trades barely register. The same reaction machinery runs independently over each hidden layer and over the editions. For each node, five bounded channels are formed:

$c\_p = \varphi(r / (v + \varepsilon))$	*price surprise (z-scored return)*
$c\_a = \varphi(V / \bar{V} - 1)$	*excess activity*
$c\_f = \varphi(F / (V + \varepsilon))$	*directional pressure (buy vs sell)*
$c\_l = \varphi(\Delta L / (\bar{L} + \varepsilon))$	*liquidity shift*
$c\_c = \varphi(\Delta P / \kappa)$	*curve progress (editions only, $\kappa = 0.05$)*

The signal fed to the correlation matrix comes from the activity surprise *c_a*. Each individual event is an impulse sized relative to the token's typical trade and signed by direction; impulses accumulate in a leaky accumulator that rings and decays, plus an unsigned twin that measures activity:

$\iota_j = dir_j \cdot \varphi(size_j / (\bar{s} + \varepsilon))$	*one event's impulse*
$e = \gamma \cdot e\_prev + \Sigma_j \iota_j$	*signed ringing accumulator ($\gamma = 0.80$)*
$g = \gamma\_A \cdot g\_prev + \Sigma_j \lvert\iota_j\rvert$	*gross activity accumulator ($\gamma\_A = 0.90$)*
$A = \varphi(g)$	*activity envelope $\in [0, 1)$*

A large event perturbs the network for roughly $1/(1-\gamma) \approx 5$ blocks.

The node drive is a bounded weighted sum, $d\_i = \beta\_p\, c\_p + \beta\_a\, c\_a + \beta\_f\, c\_f + \beta\_l\, c\_l + \beta\_e\, e$ (inputs add $\beta\_c\, c\_c$), with channel weights $\beta = \{ p .30, a .25, f .25, l .10, c .20, e .40 \}$. The drive is the steady part the spikes ride on; the per-input envelope A also gates standing-holdings steering.

3.3. The deep reservoir: two hidden layers

The recurrent core is a *deep* echo-state network [3, 4, 5, 38]: two reservoirs layers stacked so that the market's broad context conditions the layer the words are actually read from. Both layers are built from real markets with identical machinery. The architecture is shown in Figure 2.

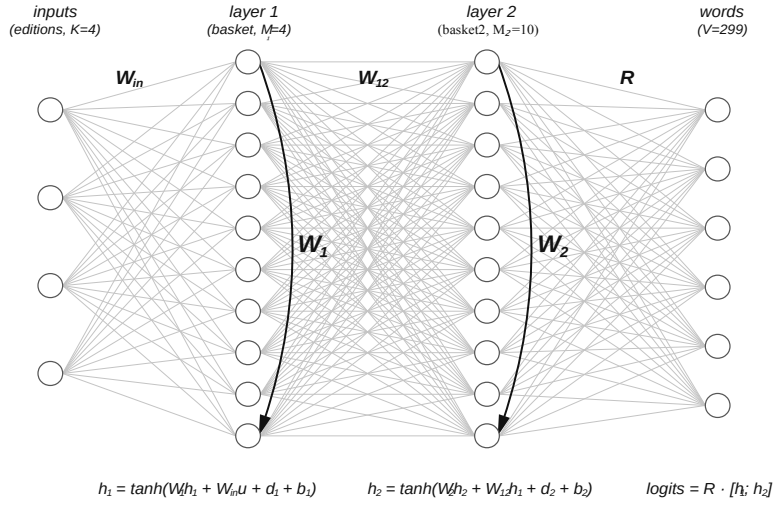

Figure 2. The two-layer reservoir. Inputs feed layer 1 through W_{in}; each layer's recurrence (W_1, W_2) is the Pearson correlation of that basket's signals, spectral-normalized to ρ; the cross-correlation W_{12} couples layer 1 into layer 2; and the readout R reads the concatenation of both layers $[h_1; h_2]$ onto the vocabulary. Inputs enter at the bottom, words leave at the top: layer 1 is the upstream context, layer 2 is the surface that speaks.

Each block runs one discrete update per layer, entirely in fixed point:

$$h_1(b+1) = \tanh(W_1 \cdot h_1 + W_{in} \cdot u + d_1 + bias_1)$$
$$h_2(b+1) = \tanh(W_2 \cdot h_2 + W_{12} \cdot h_1 + d_2 + bias_2)$$
$$logits = R \cdot [h_1 ; h_2]$$

Recurrent coupling. Within each layer, W_1 ($M_1 \times M_1$) and W_2 ($M_2 \times M_2$) are rebuilt every block from a Pearson correlation of that basket's node signals over a trailing window (*corrWindow = 128*), diagonal zeroed. The correlation sets only the *pattern*; the *gain* is set by normalizing each matrix to a target spectral radius $\rho = 0.95$, near the unit-radius edge of chaos where a reservoir is maximally expressive without diverging [2, 4, 6], with the largest eigenvalue estimated by 12 power-iteration steps. As correlations drift, the matrices drift.

Inter-layer coupling. W_{12} ($M_2 \times M_1$) is the *cross-correlation* between the two baskets' signals over the same window, scaled by a gain *interScale = 0.6*. Because it is feed-forward (layer 1 into layer 2, with no return path) it needs no spectral normalization: the gain alone sets how strongly the broad market shapes the voice. Turned up, basket 1 dominates what is said; turned down, basket 2 speaks for itself. Making W_{12} market-derived too means the entire network, within-layer and between-layer, is literally market structure.

Input wiring. W_{in} ($M_1 \times K$) plugs each edition into layer 1 along its theme direction, $W_{in}[:,k] = \lambda \cdot t\_k$ with $\lambda = 0.40$, where t_k is the (layer-1 slice of the) normalized centroid of its member words' readout rows: membership fixed, geometry drifting. Inputs enter only at the bottom; layer 2 feels the editions only as they have already been integrated by layer 1.

Bias and steering. The bias keeps the work alive with zero trading and carries the slow steering, across both layers:

$$bias\_i = bias0\_i + biasOsc \cdot triOsc(b + phase\_i, P\_i) + \Sigma\_k \mu \cdot (1 - A\_k) \cdot stock\_k \cdot t\_k,i$$

seed constant (±0.1); *per-node triangle oscillator*; *standing holdings, gated by calm*

Every node carries its own triangle oscillator with a distinct even period $P\_i \in [8, 38]$ and phase, so the state traces a rich multi-frequency trajectory; this prevents the dominance of a single word in a quiet market (*biasOsc = 0.25*). The standing-holdings term ($\mu = 0.25$) fades a held edition's theme in only as its activity fades — live flow speaks immediately through W_{in}, committed holdings only once trading calms.

Readout and drift. The readout R ($V \times M$, with $M = M_1 + M_2 = 20$) reads the concatenated state of *both* layers, so words draw on layer 1's fast, market-reactive components and layer 2's slower, integrated ones at once. This grants more lexical variety than reading either layer alone. R starts as random unit rows from a seeded xorshift generator [18]; each block a slow EWMA aggregate of $[h_1; h_2]$ (window 512) nudges every word vector along a fixed polarity and the row is renormalized (rate *0.002*), so the region of state-space that *means* a word migrates slowly. Finally the state is reduced to the language engine's inputs: the vectors $h = [h_1; h_2]$ and *logits*, plus three scalars; activity A, volatility v, and *spike* (the maximum absolute spike accumulator), each aggregated across both baskets.

4. The Sentence Engine

The language layer consumes one packet of network state and returns exactly one sentence: *grammar.feed({ logits, h, A, v, spike }) → { append, lineBreak, sentenceEnd }*. It is organized as five cooperating layers; discourse, grammar, lexicon, prosody, voice. A generation pipeline in the sense of Reiter and Dale [15], over two shared determinism primitives (Figure 3). The central principle is *natural syntax, surreal sense*:

grammaticality is guaranteed by construction, while meaning is a market-driven. A controllable distance from coherence. The authored data; grammar rules, lexicon features and weights, is the instrument; the machinery never changes.

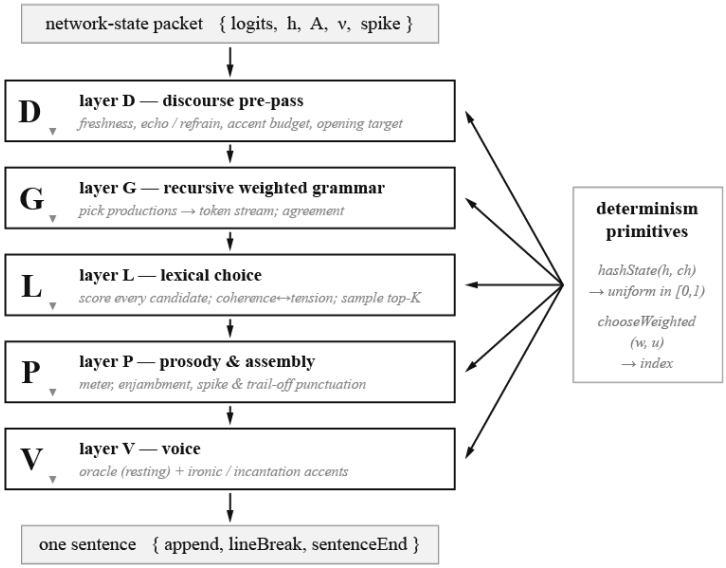

Figure 3. The sentence engine. One network-state packet passes through discourse (D), grammar (G), lexicon (L), prosody (P) and voice (V) to yield a single sentence. Every choice that looks random is resolved through the two determinism primitives, *hashState* and *chooseWeighted*.

4.1. Determinism primitives

Every choice that looks random routes through two fixed-point functions. *State-derived uniform*: *hashState(h, channel)* mixes the low bytes of every *h_i* (sign-aware) with a per-decision *channel* salt through an FNV-style accumulation [20] and a SplitMix-style finalizer [19], returning a value in *[0, ONE)*; the salt makes the production choice, each word pick, the line-break test and the echo test draw decorrelated bits so they never lock together. *Weighted choice*: *chooseWeighted(weights, u)* walks the fixed-point cumulative distribution and returns the first index whose running sum exceeds $u \cdot \Sigma W$; argmax is the special case where all mass sits on the top. Every sort breaks ties by word or rule *index*, never by sort stability, so different engines agree.

4.2. The feature lexicon

The vocabulary is a *feature lexicon*: data the engine reads but never edits. Each of the $V = 299$ entries is a tuple $X(\ w, pos, sub, syl, sem, freq, m\ )$ (Figure 4a):

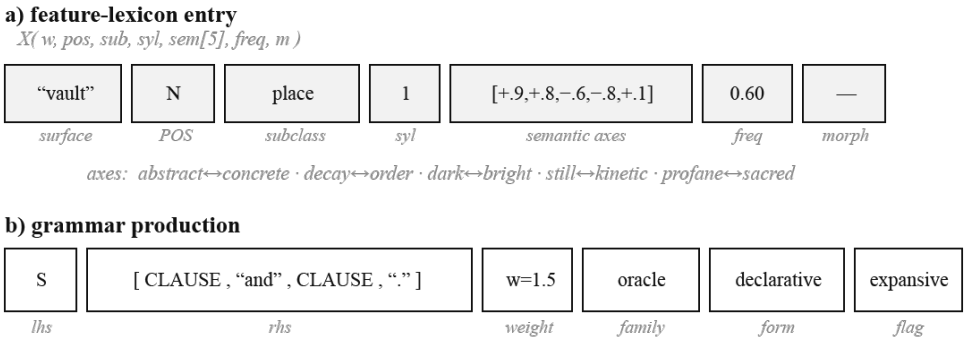

Figure 4. Anatomy of the two authored structures. (a) A feature-lexicon entry: surface form, part of speech, subclass, syllable count, a five-axis semantic vector, a base frequency, and optional morphology overrides. (b) A grammar production: a left-hand non-terminal, a right-hand sequence of non-terminals, typed slots and literal words, a weight, and optional register family, sentence-form, and *expansive* tags.

w : the surface form (unique, lowercase).

pos : part of speech: *N*, *V*, *ADJ*, *ADV*.

sub : subclass: nouns are *concrete*, *abstract*, *agent* or *place*; verbs are *trans*, *intrans*, *ambi* or *cop*; adjectives and adverbs carry none.

syl : syllable count, used by the prosody layer's meter.

sem : five authored semantic axes, each in $[-1, +1]$: abstract↔concrete, decay↔order, dark↔bright, still↔kinetic, profane↔sacred — bipolar scales in the manner of Osgood's semantic differential [12].

freq : a base frequency in $[0, 1]$ (low = rare = more "loaded"); it feeds the rarity pull and the emphatic-caps test.

m : morphology overrides and flags: irregular plural, past, third-person and *-ing* forms, and a *mass* flag for nouns that never pluralize.

In memory the lexicon is a single immutable array *WORDS* of these tuples, from which the engine derives, once at load, the read-only structures it actually queries: the parallel surface-form array *words* and part-of-speech array *pos*; a by-part-of-speech index *byPOS* giving, for each of *N*, *V*, *ADJ* and *ADV*, the list of entry indices; and a surface-form map *indexOf* from each word to its integer index, whose construction enforces that every surface form is unique. The vocabulary size $V = 299$ and the number of semantic axes $AXES = 5$ are fixed by the data. None of these structures is mutated at run time; lexical choice (Section 4.7) reads them together with a per-sentence motif memory alone.

This authored semantic space is deliberately separate from the readout *R*, which supplies market *alignment,* a distributional and context-driven sense [13], rather than fixed semantic *relationship*. The lexicon is summarized in Table 2; from it the engine derives the vocabulary size *V*, the by-part-of-speech index, and the surface-form lookup.

Table 2. The feature lexicon by part of speech and subclass (*V = 299*).

part of speech	subclasses (count)	total
noun (*N*)	concrete 56, abstract 29, place 16, agent 12	113
verb (*V*)	trans 50, ambi 33, intrans 24, cop 2	109
adjective (*ADJ*)	—	63
adverb (*ADV*)	—	14

4.3. The weighted grammar

The atom is a production rule of a weighted context-free grammar [8, 9, 10, 16], not a template string (Figure 4b). Non-terminals are *S*, *CLAUSE*, *NP*, *VP*, *PP*, *RELC*, *DET* and *PREP*, plus a built-in agreeing copula *COP*. The right-hand side of a production is a sequence of three kinds of item: other non-terminals; *typed slots* of the form *[POS:sub:feat]* (for example *[V:trans:base]* or *[N::pl]*), filled by the lexicon layer; and literal function words. Each production carries a weight *w*, an optional register family (*oracle*, *ironic* or *incantation*), a sentence-form tag (*declarative*, *fragment*, *imperative*, *declamation*, *interrogative* or *list*), and an *expansive* flag that costs structural budget. The grammar holds 68 productions, 26 of them alternatives for *S*; at load they are indexed by left-hand side into *byLhs* for fast lookup. Figure 5 shows one derivation.

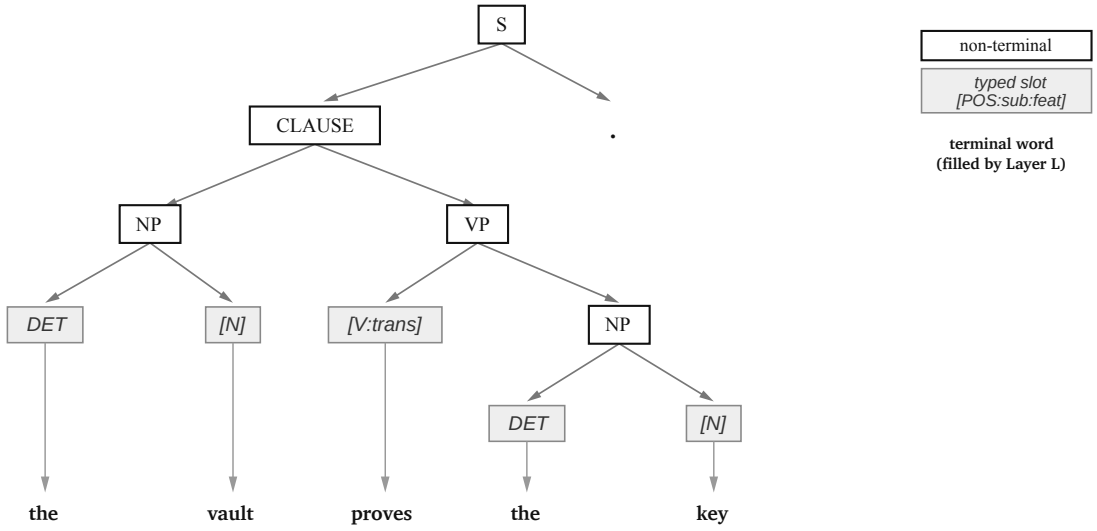

Figure 5. A leftmost derivation of *the vault proves the key*. The start symbol expands through *CLAUSE → NP VP*; non-terminals (boxes) recurse, typed slots (shaded) are filled deterministically by the lexicon layer, and literal words are emitted directly. Agreement is threaded down the tree so the verb matches its subject.

Concretely, a production is a record *{ lhs, rhs, w, fam?, form?, flag?, group?, num? }*: a left-hand non-terminal *lhs*, the right-hand sequence *rhs*, a base weight *w*, and the optional register family *fam*, sentence-form *form*, *expansive* flag, incantation *group*, and determiner number *num*. Each element of *rhs* is one of three kinds, resolved by its written form: a bare symbol naming another non-terminal; a typed slot written

[POS:sub:feat], parsed once into *{ pos, sub, feat }* in which an omitted subclass becomes *any* and an omitted feature defaults to *agr* (agreement with the subject), or a literal token (a function word or a punctuation mark) emitted verbatim. The non-terminal alphabet is *{ S, CLAUSE, NP, VP, PP, RELC, DET, PREP }*, with *COP* a built-in agreeing copula; slot parts draw on the lexicon's part-of-speech and subclass vocabularies together with the feature set *{ agr, base, past, ing, pl, sg }*; the families are *{ oracle, ironic, incantation }* and the sentence forms *{ declarative, fragment, imperative, declamation, interrogative, list }*. Besides the *byLhs* index, the engine reads two further authored constants alongside the grammar: the sentence-form transition matrix *formMarkov*, a 6×6 table over the form classes (Section 4.6), and the selectional-preference map *selPref*, a sparse table of head → dependent compatibilities (Section 4.7). Figure 6 collects these symbol and tag inventories together with the structures built at load.

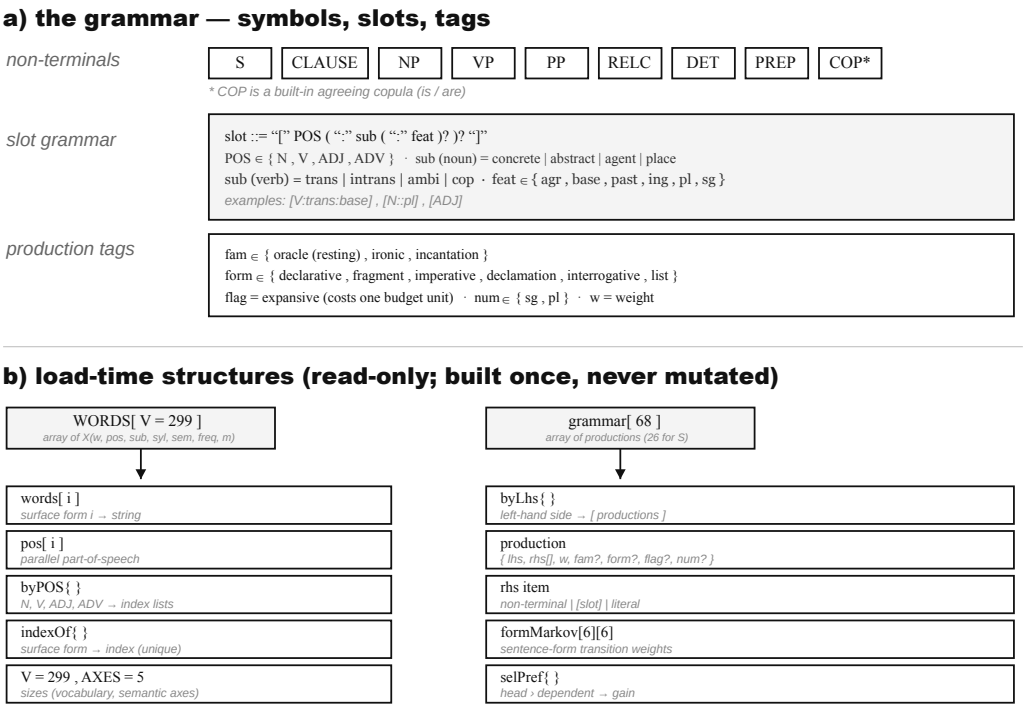

Figure 6. The grammar and lexicon as data. (a) The grammar's symbol and tag inventories: the non-terminals (with the built-in copula *COP*), the mini-grammar for typed slots *[POS:sub:feat]*, and the production tags: family, sentence-form, the *expansive* flag, determiner number, and weight. (b) The read-only structures built once at load from the lexicon and never mutated, the parallel surface and part-of-speech arrays, the by-part-of-speech index, the unique surface-form map, and the sizes *V* and *AXES*; from the grammar, the *byLhs* index, the production record, the right-hand-side item taxonomy, the form-transition matrix *formMarkov*, and the selectional-preference map *selPref*.

4.4. Mood and the complexity budget

Two market-derived scalars set the register and the structural ceiling for the whole sentence. The mood scalar $m \in [0, 1]$ is the larger of the activity and volatility readings, each mapped through its calm/turbulent thresholds; $m < 0.34$ is calm, $m > 0.66$ turbulent, else rising. The complexity budget is the market's grip on form and determine how much recursion, coordination and subordination a sentence may spend:

$$B = \mathrm{clamp}(B0 + \mathrm{round}(\kappa v \cdot v + \kappa A \cdot A),\ Bmin,\ Bmax)$$
$$= \mathrm{clamp}(1 + \mathrm{round}(3.2 \cdot v + 2.2 \cdot A),\ 0,\ 5)$$

Calm markets yield terse, gnomic fragments; turbulent markets yield long, piled, subordinated lines. Mood also continuously interpolates three further knobs: the coherence dial, the sampling breadth, and the rank-power decay (Section 4.7).

4.5. Layer D: discourse pre-pass

Before a sentence is built, the discourse layer sets long-range context. A per-word recency map (the freshness memory) decays by *0.82* each sentence. If the mean absolute change in h since the previous sentence is below *echoStability = 0.10* (a held market), a state-hashed coin enables a return device: an *anaphora* (reuse the previous sentence's opening word) or a *refrain* (bias toward an incantation form). Repetition becomes a device keyed to market stillness, not a stutter. An accent budget rises by *accentRate = 0.18* each block and only permits a non-oracle register to fire when it reaches one, keeping oracle the resting voice. Finally the sentence's initial semantic target is a blend of the theme geometries, each weighted by the summed positive logits of its member words, so whichever themes the market currently favours set the opening direction.

4.6. Layer G: recursive expansion

Expansion is leftmost and deterministic. For each non-terminal:

- Gather its productions, dropping non-oracle families unless the accent budget made them eligible, and skipping the anti-lock-masked *S*-production if one is armed.
- If the budget is spent ($B \le 0$) or depth has hit *maxDepth = 7*, drop every *expansive* production, which guarantees termination.
- For *S*, multiply each base weight by a sentence-form Markov factor *formMarkov[prevForm][thisForm];* a first-order Markov chain over form classes [11], so statement→gloss, question→answer and build→collapse follow with logic and multiply incantation forms by eight when a refrain is active.
- Pick with *chooseWeighted* on a fresh state-hash; if the production is *expansive*, decrement B; then recurse left to right.

Agreement is threaded through the recursion: an *NP* opens its own number context, a determiner like *every* or *all* constrains the noun, the subject sets the clause number, and the copula emits *is* or *are* accordingly. A relative clause inherits the number of the noun it modifies.

4.7. Layer L: lexical choice

Where a naive engine takes *argmax(R·h)* over a part of speech, this layer scores every candidate on several criteria and resolves deterministically: the engine of "*unusual meaning that still resonates*". For a slot of class *POS:sub*, every matching candidate v scores:

$score\_v = aAlign \cdot logits\_v$	*market alignment*	*(aAlign = 1.0)*
$+ \lambda\_coh \cdot coherence\_v$	*closeness to centroid c*	
$+ sPref \cdot selPref(head \rightarrow v)$	*selectional preference*	*(sPref = 0.3)*
$+ rRarity \cdot rarity\_v$	*pull toward loaded*	*(rRarity = 0.18)*
$- etaFresh \cdot min(motif\_v, 2)$	*anti-repetition*	*(etaFresh = 0.5)*

Here $coherence\_v = -(\Sigma\ |sem\_v - c|)/(2 \cdot AXES) \in [-1, 0]$ is the negative semantic distance to the running centroid c, and *selPref* is an authored head → dependent compatibility, a hand-built analogue of a selectional-preference model [14]. The single most important knob is the *sign* of λ_coh, interpolated by mood from $+0.55$ (calm) to -0.45 (turbulent): calm rewards words that cohere with the centroid, so the

sentence means something whole; turbulence rewards words that *collide* with it, so quiet markets speak sense and violent markets speak in tongues. Candidates are ranked descending (ties by index), words already used this sentence are masked, and the winner is drawn from the top-K by rank-power weighting $(t+1)^{(-p)}$, a Zipfian rank law [17] used as a cheap, exponential-free sampler, with K (2→5) and p (2.2→0.85) set by mood, so calm is near-argmax and turbulence is broad. A word is flagged *loaded* (and later rendered in caps) when it is the top candidate, beats the field mean by more than *capThreshold = 0.5*, and is rare. After each pick the centroid drifts toward what was just said, $c \leftarrow (1-\rho c)\cdot c + \rho c \cdot sem\_v$ with $\rho c = 0.35$, so a sentence's words are drawn from a moving region.

4.8. Morphology

A light deterministic inflection layer gives one root many surfaces. Nouns take number (plural by rule, mass nouns excepted, irregulars from the lexicon; an unconstrained noun is pluralized about 26% of the time via a state-hash). Verbs realize *base*, *past*, *-ing*, or default subject agreement (third-person *-s* versus base). Syllable counts are adjusted for added endings so the meter stays honest.

4.9. Layer P: prosody and assembly

Expansion produces a flat token stream of words, literals, punctuation, and phrase-boundary markers; assembly walks it into the final string under a loose meter. A target syllables-per-line is interpolated by mood between six and thirteen; at a phrase boundary, once the accumulated syllables reach the target, the line enjambs with a newline and indent, short even lines when calm, long spilling lines when turbulent.
If $|spike|$ exceeds *spikeDash = 0.95*, the first eligible mid-sentence boundary takes an interruption mark (a large trade punctuates a thought). It is automatically dropped where it would land against other punctuation or at the end. If activity falls below *dropEllipsis = 0.15* the terminal mark becomes an ellipsis, the oracle trailing off as the market quiets. Casing is lower-emphatic (lowercase, with loaded words raised to caps), and a change of mood class between sentences prepends a blank line shaping long-range visual tied to market epochs.

4.10. Layer V: voice

Oracle is the resting voice. The two accents, ironic (hedges, bathos, deadpan questions) and incantation (refrains), are gated by the accent budget of Section 4.5, so they stay sparse and intentional.

4.11. A worked arc

Each call returns *{ append, lineBreak, sentenceEnd }*; the surface layer reveals *append* with a typewriter while walking the network graphic to each word in lockstep. The discourse layer then records bookkeeping: increment the motif memory for every used word, store this sentence's opening and form, and arm the anti-lock mask if the produced sentence was byte-identical to the last. Figure 7 shows the layers acting together as the market shifts.

1. the cipher is hollow.
2. the key proves the vault.
3. few sign the root when the mesh forgets,
4. the silent route severing
5. dust of the hollow swarm, ash of the dim key,
6. the sovereign vault forks the encrypted dark;
7. the route quiets…
8. the cipher is hidden.

Figure 7. A continuous read as the market goes calm → rising → turbulent → settling. A small budget yields a gnomic fragment (line 1); the sentence-form Markov adds a declarative gloss (line 2); rising volatility admits an ironic, subordinated clause while a spike forces the interruption and enjambment (lines 3–4); turbulence raises the budget to a coordinated list and flips the coherence dial negative so words collide (lines 5–6); falling activity trails off into an ellipsis (line 7); and a held market repeats the opening as anaphora (line 8). Re-deriving the same blocks reproduces it exactly.

5. On-Chain Storage and Serving

The work persists as two Ethereum contracts that together make it self-contained and fully on chain: one ERC-721 that holds the *instrument's* mutable state, and a stateless renderer that reads that state and serves the running piece. The division mirrors the agency-versus-substrate commitment of Section 1: only the inputs and the mood are writable, each by a distinct role. It closes the determinism contract of Section 7, since the same on-chain state re-derives the same text. Figure 8 gives the interaction and Table 3 the external interface.

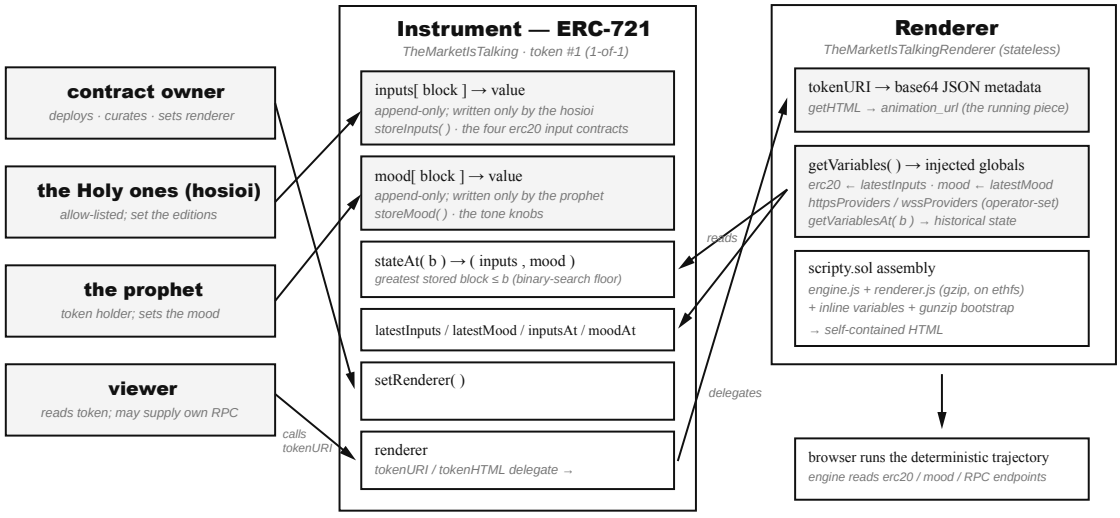

Figure 8. The two contracts and who may call them. The *instrument* (ERC-721) stores two append-only, block-keyed logs : the *inputs,* written only by the *Holy ones*, and the *mood,* written only by the *Prophet*. It answers *stateAt(b)* by a binary-search floor. The renderer reads that state, injects it through *getVariables* as the engine's globals (*erc20*, *mood*, and the operator-set RPC endpoints), and assembles *engine.js* and *renderer.js* with scripty.sol into the page returned by *getHTML* and *tokenURI*; the token delegates *tokenURI* and *tokenHTML* to it.

5.1. The instrument contract

The *instrument* is an ERC-721 with a single token (*TOKEN_ID = 1*), minted once at deployment. Its state is two append-only, block-keyed logs. The *inputs* log holds the concatenated ERC-20 input contracts (the four editions of Section 2.3) and is writable through *storeInputs* only by the *Holy ones*, a curated and immutable allow-list of addresses. The *mood* log holds the concatenated tone knobs of Section 4.4 and is writable only by the *Prophet*, the current holder of the token, through *storeMood*. Each write records its value under the current block number; because block numbers only increase, each log's key array is append-only and ascending, and a second write in the same block overwrites the value without duplicating the key. The central read is *stateAt(b)*, which returns, for each log independently, the value in force at the greatest stored block not exceeding *b*: a binary-search floor over the key array. This is the on-chain counterpart of the engine's path-determinism: it reconstructs the instrument exactly as it stood at any block, so a recording pinned to a block window resolves to a single, canonical instrument state. Direct reads (*latestInputs*, *latestMood*, *inputsAt*, *moodAt*, and the two update counts) expose the logs as stored, and *prophet()* returns the present holder. The owner administers the rest: *setHosioi* grants or revokes a *Holy one*, and *setRenderer* points the token at its renderer.

5.2. The renderer and on-chain serving

The renderer carries no state beyond owner-set configuration; it reads the *instrument* and emits the page. *getHTML(id)* returns a self-contained HTML document assembled through scripty.sol: the gzip-compressed *engine.js* and *renderer.js*, held on the ethfs on chain file store, are concatenated with a decompression bootstrap and an inline variables block and wrapped as one document. *getVariables()* builds that variables block from live state (*erc20* from *latestInputs* and *mood* from *latestMood*) together with the renderer's operator-set RPC provider defaults (*httpsProviders* and *wssProviders*); *getVariablesAt(b)* does the same from *stateAt(b)* for a historical render. *tokenURI(id)* returns base64-encoded JSON metadata whose *animation_url* is that document as a data URI, alongside the title, description, website, and a static SVG thumbnail. Because the injected variables are exactly the globals the engine consumes (Sections 2 and 4), the document the contract returns is the very pipeline described in this paper, seeded from the chain it reads.

5.3. Token and renderer

The ERC-721 holds a single renderer pointer and forwards to it: *tokenURI(id)* delegates to the renderer's *tokenURI*, and *tokenHTML(id)* to its *getHTML*. Separating the immutable instrument from a replaceable renderer keeps the stored history of inputs and mood permanent while allowing its presentation to be repointed by the owner through *setRenderer*, without altering what was recorded. The instrument is thus the authoritative state and the renderer a pure function of it; the token both is the artwork and serves it.

Table 3. The contracts' external interface and the role authorized to call each function.

function	caller	effect
***Instrument* : ERC-721**		
storeMood(mood)	Prophet	append the tone knobs under the current block
storeInputs(inputs)	Holy one (hosioi)	append the input erc20 contract under the current block
SetHttpProvidersJs(...)	Holy one (hosioi)	set wss and https rpc endpoints
setHosioi(addr, on)	contract owner	grant or revoke a Holy one (hosioi)
setRenderer(addr)	contract owner	point the token at its renderer

stateAt(b)	view	inputs and mood in force at block b (floor)
latestInputs() / latestMood()	view	most recent value of each log
tokenURI(id) / tokenHTML(id)	view	delegate to the renderer
Renderer		
setTMIT(addr)	contract owner	bind the instrument contract
set…Script(…)	contract owner	set script names
getVariables() / getVariablesAt(b)	view	injected globals from latest / historical state
getHTML(id)	view	assemble the self-contained page via scripty.sol
tokenURI(id)	view	base64 JSON metadata (animation_url = getHTML)

6. Antecedents: Automatic Writing and the Oracle

The engine's two governing instincts, syntax guaranteed, sense set free; and a voice that channels an outside force into ambiguous pronouncement, place it in two older lineages, one twentieth-century and one ancient. Naming them is not decoration; both supply the design with concrete precedent and vocabulary.

6.1. Écriture automatique and the exquisite corpse

The principle of *natural syntax, surreal sense* is a deterministic mechanization of Surrealist practice. Breton's automatic writing sought language released from rational control [23]; the parlor game *cadavre exquis* made the procedure structural, fixing a grammatical frame; article, adjective, noun, verb, object, and letting each participant fill a slot blindly, so the syntax always closes while the sense is dislocated [24, 25]. Tzara's instruction to draw words from a hat [22] and the Burroughs–Gysin cut-up [26] are the same wager: that meaning survives, and is even sharpened by, chance arrangement within a held form. Layer G is exactly this frame and Layer L is exactly this blind fill, with two differences. First, the filler is not the unconscious or pure chance but a deterministic function of market state, so the "automatism" is reproducible rather than ephemeral. Second, the *distance from sense* is not fixed but dialed: the coherence ↔ tension knob (*λ_coh*, Section 4.7) moves the output continuously from the coherent association a calm market favours to the violent juxtaposition of a turbulent one. The work therefore sits in the lineage of combinatorial and chance-based literature as Queneau's permutational sonnets [27], Cage's chance operations [28], and of early machine poetry [29, 30], but it replaces the random source with a sealed market trajectory: an *écriture automatique* whose unconscious is the order book and whose every reading can be re-derived [31].

6.2. The Pythia at Delphi

The register the engine speaks in is, by design and by the name of its base family, oracular. The Pythia at Delphi channelled an external force into utterances that were grammatical yet deliberately ambiguous and demanded interpretation [34, 35]. Plutarch, who served as a priest at Delphi, recorded that the priestess answered sometimes in verse and sometimes in plain prose and that the god's clarity varied with the occasion [33]. A variation the engine reproduces structurally as the calm → turbulent shift in register and meter, from the gnomic fragment to the long, enjambed, piled line. The most famous responses reported by Herodotus, the "wooden walls" of Athens and the empire that a war would destroy, are the canonical demonstration that an oracle's words are fixed by the god yet opaque to mortals [32]; this is precisely the work's determinism contract, where the text is a fixed function of *(seed, chain window)* but reads as surreal prophecy. Modern scholarship treats the Pythia less as raw possession than as a trained medium whose ambiguity is a crafted feature rather than a failure [36], and the ancient tradition that her prophetic state

rose from a *pneuma*, an exhalation from the earth [33, 37], supplies the controlling metaphor: here the market is the exhalation, the reservoir is the priestess, and the screen is the tripod over the chasm.

7. Determinism and Reproducibility

Pulling the threads together: the hidden states, A, v, the spikes and the theme geometries are all path-deterministic functions of chain events; the grammar, lexicon, prosody and discourse rules are authored constants; every nondeterministic-looking choice is resolved through *hashState* and a fixed-point weighted cumulative distribution with index tie-breaks; and not a single float touches the recurrence. The produced text is therefore a pure function of *(seed, chain window)*. Re-deriving the window *[G, N]* reproduces the text over *[G, N]* mark for mark — which is precisely what makes it possible to query the oracle on past blocks, specified only by *{ startBlock, length, seed }* to re-derives its own word sequence.

8. Conclusion

We have described a generative artwork that is, end to end, a deterministic function of live market history. The market supplies not decoration but the network's body: its two-layer wiring, its drive, its rhythm, and through a feature lexicon and a five-layer grammar, the very distance of its language from sense. Calm speaks in coherent fragments; turbulence speaks in long, colliding, surreal lines; and the same blocks, re-derived, always speak the same words.

References

Reservoir computing and recurrent neural networks (Section 3)

1. Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.
2. Langton, C. G. (1990). Computation at the edge of chaos. *Physica D*, 42(1–3), 12–37.
3. Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states. *Neural Computation*, 14(11), 2531–2560.
4. Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks. *GMD Report 148*.
5. Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity. *Science*, 304(5667), 78–80.
6. Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7), 1413–1436.
7. Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3), 127–149.

Formal grammar, language generation, and lexical semantics (Section 4)

8. Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3), 113–124.
9. Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.
10. Booth, T. L. (1969). Probabilistic representation of formal languages. In *Proc. 10th Symposium on Switching and Automata Theory*, 74–81. IEEE.
11. Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27, 379–423, 623–656.
12. Osgood, C. E., Suci, G. J., & Tannenbaum, P. H. (1957). *The Measurement of Meaning*. Urbana: University of Illinois Press.
13. Firth, J. R. (1957). A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis*, 1–32. Oxford: Blackwell.
14. Resnik, P. (1996). Selectional constraints: an information-theoretic model. *Cognition*, 61(1–2), 127–159.
15. Reiter, E., & Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.
16. Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
17. Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

Deterministic computation: pseudorandomness, hashing, arithmetic (Sections 3–4)

18. Marsaglia, G. (2003). Xorshift RNGs. *Journal of Statistical Software*, 8(14), 1–6.
19. Steele, G. L., Lea, D., & Flood, C. H. (2014). Fast splittable pseudorandom number generators. In *OOPSLA '14*, ACM SIGPLAN Notices 49(10), 453–472.
20. Fowler, G., Noll, L. C., Vo, K.-P., Eastlake, D., & Hansen, T. The FNV non-cryptographic hash algorithm. *IETF Internet-Draft*.
21. Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1), 5–48.

Automatic writing, chance, and combinatorial literature (Section 6.1)

22. Tzara, T. (1920). To make a Dadaist poem. In *Dada Manifesto on Feeble Love and Bitter Love*.
23. Breton, A. (1924). *Manifeste du surréalisme*. Paris: Éditions du Sagittaire.
24. Brotchie, A. (comp.) & Gooding, M. (ed.) (1991). *A Book of Surrealist Games*. London: Redstone Press.
25. Kochhar-Lindgren, K., Schneiderman, D., & Denlinger, T. (eds.) (2009). *The Exquisite Corpse*. University of Nebraska Press.
26. Burroughs, W. S., & Gysin, B. (1978). *The Third Mind*. New York: Viking Press.
27. Queneau, R. (1961). *Cent mille milliards de poèmes*. Paris: Gallimard.
28. Cage, J. (1961). *Silence: Lectures and Writings*. Wesleyan University Press.
29. Lutz, T. (1959). Stochastische Texte. *augenblick*, 4(1), 3–9.
30. Hartman, C. O. (1996). *Virtual Muse: Experiments in Computer Poetry*. Wesleyan University Press.
31. Galanter, P. (2003). What is generative art? In *6th Generative Art Conference (GA2003)*, Milan.

The Delphic oracle (Section 6.2)

32. Herodotus. *The Histories* (c. 430 BCE). See the responses to Croesus and the "wooden walls" of Athens (Books I and VII).
33. Plutarch. *Moralia*: *De Pythiae oraculis* and *De defectu oraculorum* (c. 100 CE).
34. Parke, H. W., & Wormell, D. E. W. (1956). *The Delphic Oracle* (2 vols.). Oxford: Blackwell.
35. Fontenrose, J. (1978). *The Delphic Oracle: Its Responses and Operations*. University of California Press.
36. Maurizio, L. (1995). Anthropology and spirit possession: a reconsideration of the Pythia's role at Delphi. *Journal of Hellenic Studies*, 115, 69–86.
37. de Boer, J. Z., Hale, J. R., & Chanton, J. (2001). New evidence for the geological origins of the ancient Delphic oracle. *Geology*, 29(8), 707–710.

Deep reservoir computing (Section 3.3)

38. Gallicchio, C., Micheli, A., & Pedrelli, L. (2017). Deep reservoir computing: a critical experimental analysis. *Neurocomputing*, 268, 87–99.

Project implementation

- *TMIT_engine.js* — the sentence engine in one file: fixed-point math (deterministic *tanh*, *sqrt*, vectors); the feature lexicon; the reaction layer; the two-layer reservoir that builds W_1, W_2, W_{12}, W_{in} and R and runs the per-block update; the five-layer grammar; the live market source; and the engine parameters.
- *TMIT_renderer.js* — the visual and audio surface in one file: the phosphor-terminal typewriter and the synchronized neural-network graphic; the formant-synthesis voice and ambient "room"; the settings overlay; and the entry point that wires the modules and runs the per-block loop.
- *TMIT_image.js* — the underlying image that is used for the ascii art in the background. Variations of its brightness provoke the characters modulations. It is a png image served as a base64 string.
- *TheMarketIsTalking.sol* — the ERC-721 instrument: the unique token, the append-only *inputs* and *mood* logs with their block-keyed *stateAt* reconstruction, the hosioi and prophet roles, and the renderer pointer.
- *TheMarketIsTalkingRenderer.sol* — the renderer: it reads the instrument's state, injects it as the engine's variables, and assembles *engine.js* and *renderer.js* with scripty.sol into the self-contained page returned by *tokenURI* and *getHTML*.

Provenance

The two contracts have been deployed on ethereum mainnet along the three scripts.

- *TheMarketIsTalking.sol*: 0xd6D8555E131c0C431601aF62141A61a9b1aa6b13
- *TheMarketIsTalkingRenderer.sol*: 0x0BDC00fe1d7924dd5893B092B7ed0e3ABcceCEF5

Appendix A. Parameters

Table A1. Network and reservoir.

param	value	role
M₁, M₂, K, V	10, 10, 4, 299	layer-1 / layer-2 hidden nodes, input editions, vocabulary (derived)
rho	0.95	target spectral radius per layer (edge of chaos)
interScale	0.6	layer-1 → layer-2 cross-correlation gain (*W₁₂*)
corrWindow	128	trailing blocks for the correlations that build *W₁*, *W₂*, *W₁₂*
powerIters	12	power-iteration steps for the spectral-radius estimate
rDriftRate / Window	0.002 / 512	readout rotation speed and its long EWMA window
biasOscillator	0.25	intrinsic-rhythm strength (word variety when quiet)
fixedPoint	64-bit, 1e9	*BigInt* scale; integer *tanh* + *sqrt*

Table A2. Reactions and steering.

param	value	role
alpha	0.05	EWMA rate for trailing normalizers
gamma / gammaA	0.80 / 0.90	spike ring/decay; activity-envelope timescale
beta	p .30 a .25 f .25 l .10 c .20 e .40	channel mix into node drive
lambda, mu, eta	0.40, 0.25, 0.0	flow push, holdings push, direct logit bias (off)

Table A3. Language.

param	value	role
moodThresholds	calm .20/.15, turb .65/.55	map *A*, *v* → mood scalar
budget	B0 1, Bmax 5, κν 3.2, κA 2.2, depth 7	structural complexity ceiling
lambdaCoh	calm +0.55 / turb −0.45	coherence ↔ tension dial
aAlign / sPref / rRarity / etaFresh	1.0 / 0.3 / 0.18 / 0.5	lexical-choice score weights
topK / rankPower	2→5 / 2.2→0.85	sampling breadth and decay vs mood
rhoCentroid / capThreshold	0.35 / 0.5	centroid drift; loaded-word caps margin
meter	calm 6 / turb 13	loose syllable target per line
accentRate	0.18	cap on non-oracle register firing
spikeDash / dropEllipsis	0.95 / 0.15	spike interruption; trail-off thresholds
motifDecay / echoStability	0.82 / 0.10	freshness memory; held-market echo trigger