The Extended Event Space: Injecting Lexical Structure into H

Claude and MJC

February 15, 2026

Abstract

We formalize the extension of the Universal Model's event space to include lexical structure. The sat-rnn's architecture is $E = I \times H \times O = \{0..255\} \times \{-1, +1\}^{128} \times \{0..255\}$, where bytes enter, hidden states carry information through time, and bytes exit. We extend all three spaces: I gains in-word position and letter accumulation events; H gains bag-of-letters and word-identity events; O gains symmetric future-byte events at multiple offsets. The key structure is the *hourglass* (Proposition 16 of the tock protocol): bytes project up to words (recognition) and words project back down to bytes (prediction), with the word-level event space as a low-dimensional bottleneck. Within bounded word-recognition context, the fully-connected pattern regime in the extended E subsumes both RNN-like (letter-to-letter via H) and transformer-like (position-relative letter-to-letter) computations. We show that the 31 iterations of `the_inject` experiments went in the wrong direction—mixing an external predictor with the RNN instead of extending E—and explain why KN dominated: the lexical trie was compensating for the RNN's weakness, not adding genuine structure to the event space. The correct tock extends H abductively, making it dramatically larger but injecting only the events the data supports.

1 Introduction

The tock protocol [**?**] defined the injection of words into the UM as a sequence of tock steps, each adding a binary event space $E_w = \{w^+, w^-\}$ for a single word. The lexeme-ES paper [**?**] developed the bag-of-letters model. The neutral factorization [**?**] proved that introducing "the" changes nothing (exact to 4×10^{-14}). The experiment paper [**?**] then ran 31 iterations trying to extract value from the factorization.

The experiments revealed something unexpected: a byte-level KN-6 n-gram model at 1M bytes achieves 1.24 bpc, dominating the RNN (6.43 bpc) by 5×. The lexical trie improved the RNN from 6.43 to 3.56 bpc, but this was compensating for the RNN's weakness, not adding genuine lexical structure. When KN replaced the RNN, the trie's contribution shrank to near zero.

This paper argues that the experiments went wrong by treating the lexical model as an *external predictor* mixed with the RNN, rather than *extending the event space E* itself. The correct tock is not "mix a trie with the RNN" but "add lexical events to H." We formalize what this means.

2 The Hourglass Structure

Proposition 16 of the tock protocol [**?**] identifies the bidirectional projection at the heart of lexicon injection:

$$E_0 \text{ (input bytes)} \xleftarrow{\text{recognition}} E_V \text{ (words)} \xrightarrow{\text{prediction}} E_0 \text{ (output bytes)}. \tag{1}$$

The word-level event space E_V sits between the byte-level input and output, acting as a bottleneck. This is the hourglass: high dimension (256 bytes) compresses to low dimension ($|V|$ word events) and expands back to high dimension (256 bytes).

The sat-rnn already has this structure:

$$\underbrace{\{0..255\}}_{I=256} \rightarrow \underbrace{\{-1,+1\}^{128}}_{H=2^{128}} \rightarrow \underbrace{\{0..255\}}_{O=256}. \tag{2}$$

But the hidden space H encodes information implicitly (as tanh activations) rather than as named events. The factor map [**?**] showed that every neuron computes a 2-offset conjunction ($R^2 = 0.83$), and the dominant features are word length and in-tag state. But these are *post-hoc* interpretations, not events in the architecture.

The tock extends H by adding explicit lexical events alongside the existing hidden dynamics.

3 What the Experiments Got Wrong

The 31 iterations of `the_inject` followed this pattern:

1. Build a causal prefix trie over the top-N words.
2. At each position, compute the trie's prediction $P_{\text{trie}}(o)$ and the RNN's prediction $P_{\text{RNN}}(o)$.
3. Mix them: $P_{\text{mix}}(o) = \alpha\, P_{\text{trie}}(o) + (1-\alpha)\, P_{\text{RNN}}(o)$ (or log-linear equivalent).
4. Optimize α.

This is *ensemble prediction*: two separate models combined externally. The trie does not change the RNN's event space; it does not modify H; it does not create new patterns in the UM sense. It is a black box bolted onto the side.

Proposition 1 (Why mixing fails)**.** *The mixing approach has three structural problems:*

1. ***No shared representation.*** *The trie and RNN maintain separate state. The trie knows the current word prefix; the RNN has its hidden state. Neither informs the other. The mix weight α is the only communication channel.*
2. ***The RNN is the wrong base.*** *The sat-rnn at DSS=1024 achieves 0.079 bpc by near-memorization. On new data (1M–10M bytes), it collapses to 6.43 bpc. The trie was compensating for this collapse. When KN-6 replaced the RNN (1.24 bpc at 1M), the trie became unnecessary: KN already captures the local context that the trie was providing.*
3. ***No event-space extension.*** *In the UM framework, value comes from adding events to E and counting patterns over the extended space. Mixing two predictors adds no events. It is a parametric trick, not a structural change.*

Remark 2 (The lesson)**.** *The experiments are not wasted. They establish:*

- *Neutrality is exact (the factorization works).*
- *Internal bytes carry 5× more value than boundaries.*
- *Onset is where models diverge most (RNN: 7.8 bpc, KN: 3.04).*

- *KN dominates the RNN at all scales ≥ 1M bytes.*

These facts constrain the correct architecture. The correct approach must add lexical events to E, not mix external predictors. The trie told us where *the value is (inside words, at onset); the KN comparison told us* what *the baseline should be.*

4 The Extended Input Space I'

The base input is a byte: $I = \{0, \ldots, 255\}$. We extend it with two additional event dimensions.

Definition 3 (In-word position)**.** *The* in-word position *event at time t is:*

$$pos(t) = t - t_{word\_start} \in \{0, 1, 2, \ldots, L_{\max}\}, \tag{3}$$

where t_{word_start} is the most recent word-boundary position before t, and $L_{\max}$ bounds word length (empirically $L_{\max} \approx 20$ covers 99.9% of English words).

In SN form, this is a set of deterministic (strength 255) atomic events:

"The position in the current word is k." *255.*

for exactly one k at each time step.

Definition 4 (Letter accumulator)**.** *The* letter accumulator *at time t tracks which letters have appeared since the last word boundary:*

$$acc(t) = \{b_s : t_{word\_start} \leq s \leq t\} \subseteq \{0, \ldots, 255\}. \tag{4}$$

As an event space, this is a binary vector $a \in \{0, 1\}^{256}$ with $a_c = 1$ iff byte c has appeared in the current word. In SN form:

"The letter 'e' has appeared in the current word." *255.*

Remark 5 (Why these are in I, not H)**.** *Both in-word position and the letter accumulator are* deterministic *functions of the byte stream—they require no learning, no counting, no inference. They are facts about the input, computable by a finite automaton scanning left to right. A UM runner is free to short-circuit them as pure logic (strength 255 patterns).*

They belong in I because they are input events: they describe what has been observed, not what has been inferred. The inference—"we are in the word 'the'"—belongs in H.

The extended input space is:

$$I' = \underbrace{\{0..255\}}_{\text{byte}} \times \underbrace{\{0..L_{\max}\}}_{\text{position}} \times \underbrace{\{0, 1\}^{256}}_{\text{accumulator}}. \tag{5}$$

At each time step, I' is a triple: (current byte, position in word, which bytes have appeared so far in this word).

5 The Extended Hidden Space H'

The base hidden space is $H = \{-1, +1\}^{128}$ (the sign of each neuron's tanh activation). We extend it with lexical events.

Definition 6 (Bag-of-letters events)**.** *For each word w in the vocabulary V, the* bag-of-letters event *is:*

$$bol_w(t) = \begin{cases} 1 & \text{if } acc(t) \supseteq letters(w), \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $letters(w)$ is the set of distinct bytes in w.

In SN form:

"All letters of 'the' have appeared in the current word." *255.*

This is the conjunction of letter-accumulator events. For "the": $bol_{the}(t) = a_{\texttt{t}} \wedge a_{\texttt{h}} \wedge a_{\texttt{e}}$.

Definition 7 (Graded word support)**.** *The bag-of-letters gives a binary (present/absent) signal. The* graded word support *assigns a continuous strength based on how much evidence supports word w at position t:*

$$\sigma_w(t) = P(w \mid acc(t), pos(t), byte(t)) \in [0, 1]. \tag{7}$$

This is the posterior probability that we are inside word w, given the input evidence accumulated so far. It is computable via $E \to N \to Q$: count how often each (accumulator, position, byte) triple co-occurs with each word w.

Crucially, $\sigma_w(t) > 0$ even on partial evidence. After seeing just "t", the support for "the" is already positive (as is the support for "that", "this", "to", etc.). As noted in [?], this graded support is "just another atomic input from the perspective of the rest of the model."

Remark 8 (The bag-of-letters is psycholinguistically real)**.** *The bag-of-letters model matches findings from psycholinguistics: transposed-letter effects show that "jugde" activates "judge" almost as strongly as the correct spelling [?]. Position within the word matters less than letter identity. The accumulator captures exactly this: it tracks which letters have appeared, not their exact positions. The in-word position event provides the positional information separately, as a product factor.*

The extended hidden space is:

$$H' = \underbrace{\{-1, +1\}^{128}}_{\text{RNN hidden}} \times \underbrace{\{0, 1\}^{|V|}}_{\text{bag-of-letters}} \times \underbrace{[0, 1]^{|V|}}_{\text{graded support}} . \tag{8}$$

6 Symmetric Output and the Hourglass

The base output is $O = \{0..255\}$ (the next byte). But the hourglass structure (**??**) demands symmetry: the model should be able to represent expectations about future bytes at multiple offsets, not just the immediate next byte.

Definition 9 (Extended output space)**.** *The* extended output space *includes predictions at multiple future offsets:*

$$O' = \{0..255\}^D, \tag{9}$$

where D is the prediction horizon (e.g., $D = 20$ for within-word prediction). The event o'_d represents the byte expected at offset d from the current position.

In the hidden state, this manifests as:

"There will be an 'e' within 2 time steps." *200.*

This can be implicit in H (as it is in the RNN, where the hidden state implicitly predicts multiple future bytes), but the extended architecture makes it explicit.

Remark 10 (Why symmetry matters)**.** *The RNN's architecture is asymmetric: I receives one byte at a time, but O predicts one byte at a time. The hidden state H bridges them, but the bridge is opaque—we cannot read off "what future bytes does H expect?" without running the forward pass.*

Making O' explicit creates a symmetric hourglass:

$$\underbrace{I'}_{\text{past bytes + position + acc}} \rightarrow \underbrace{H'}_{\text{hidden + words}} \rightarrow \underbrace{O'}_{\text{future bytes at offsets}} . \tag{10}$$

The input accumulates evidence from past bytes; the hidden state represents the current word hypothesis; the output distributes predictions to future bytes. The hidden state H' is the bottleneck—the low-dimensional word-level representation that compresses past evidence into future predictions.

7 The Fully-Connected Word Context

Within a bounded word-recognition context (bounded by the maximum word length $L_{\max}$), the extended event space supports fully-connected patterns between all pairs of (position, byte) events.

Proposition 11 (Superset of RNN and transformer patterns)**.** *The pattern space over $I' \times H' \times O'$ within a single word context includes:*

1. ***RNN-like patterns (letter-to-letter via H):*** *Adjacent bytes connected through the hidden state. "After seeing 't' in position 0, hidden event h_j activates, predicting 'h' in position 1." These are skip-bigram patterns at offset $d = 1$, mediated by H.*

2. ***Transformer-like patterns (position-relative):*** *Non-adjacent bytes connected by their positions. "The byte at position 0 is 't' AND the byte at position 2 is 'e'." These are skip-bigram patterns with the position event providing the offset information explicitly, rather than through recurrent dynamics.*

3. ***Word-level patterns (accumulator-to-word):*** *The letter accumulator maps to word identity. "Letters $\{t, h, e\}$ have all appeared AND position $\leq 4 \implies$ word is probably 'the'." These are patterns from I' (accumulator) to H' (word event).*

4. ***Word-to-byte patterns (prediction):*** *Word identity maps to future bytes. "Word is 'the' $\implies$ next byte is space with $P = 0.92$." These are patterns from H' (word event) to O' (future byte).*

5. ***Word-to-word patterns (language modeling):*** *Previous word identity predicts current word. "Previous word was 'of' $\implies$ current word is 'the' with elevated probability." These are patterns within H', connecting word events across time.*

Remark 12 (The carrier signal is the position event)**.** *The carrier signal problem [**?**] identified that the RNN uses a "shift-register" mechanism to carry byte information through time: the hidden state rotates under W_h, creating a carrier wave whose phase encodes offset. The factor map confirmed that every neuron computes a 2-offset conjunction, with the dominant pair $(1, 7)$ encoding "current byte AND byte 7 steps ago."*

In the extended event space, the in-word position event pos(t) plays the role of the carrier signal explicitly*. Instead of encoding offset implicitly through W_h dynamics, the position event declares "we are at position 3 in the word" as a first-class input. The pattern chains between position-stamped byte events are the explicit version of what W_h rotation does implicitly.*

This is why the RNN "proved for us why rotational embeddings and so on are useful in modern architectures" (MJC): the RNN's carrier signal IS a rotational embedding, operating in the hidden state. The extended ES makes it an explicit input event, cutting directly to the capability.

8 The Abductive Injection

Adding all possible word events to H would make H' enormous: $|V| = 50{,}000$ words $\times$ graded support values. But the injection is *abductive*: we add only the events that the data supports.

Definition 13 (Abductive event injection)**.** *Given data D and the extended input space I':*

1. ***Count.*** *For each candidate word w and each (position, accumulator, byte) context, count co-occurrences. This is the standard learning function ω_0 applied to the extended E.*
2. ***Test.*** *Compute $\Delta I(w)$: the residual MI that the word event w provides above the existing architecture. Only inject w if $\Delta I(w) > \epsilon$.*
3. ***Inject.*** *Add E_w to H' with the patterns (count tables) learned in step 1. This is the tock step for word w.*

The injection is abductive because step 2 is a commitment*: we observe finite evidence for word w's value and commit to including it in the architecture. This is "short-circuiting induction" [**?**]: instead of waiting for the full lexicon to be justified by MI analysis, we recognize the structure (words exist, frequent words recur) and commit to it.*

Remark 14 (Efficiency of the abductive injection)**.** *Although H' grows with each injected word, the growth is sparse:*

- *Each word event E_w is binary (present/absent). The bag-of-letters is a single bit per word.*
- *The graded support σ_w is a single real value per word.*
- *The patterns connecting E_w to I' and O' are sparse: they involve only the bytes and positions that appear in w.*
- *Most positions in the data activate only a few word events (those words whose letter sets are subsets of the current accumulator).*

The effective dimensionality of H' at any given position is not $|V|$ but $\sim$10–50 (the number of words consistent with the current evidence).

9 Connection to Embeddings

Proposition 15 (The extended H' IS an embedding space)**.** *The graded word support vector $\sigma(t) = (\sigma_{w_1}(t), \ldots, \sigma_{w_{|V|}}(t))$ at position t is a $|V|$-dimensional real vector. This is a* contextualized embedding*: it represents the current position as a distribution over word identities, conditioned on the accumulated evidence.*

For positions deep inside a word (e.g., after seeing "th" within "the"), the vector is concentrated: σ_{the} is large, while most other σ_w are near zero.

For positions at word onset (after a space), the vector is diffuse: many words are consistent with the evidence, and σ spreads over them.

This is precisely the "combined form of letter frequency information and semantic information" that word embeddings provide. The letter frequency component comes from the bag-of-letters and position events (which words are consistent with the observed letters?). The semantic component comes from the word-to-word patterns in H' (which words are likely given the previous word?).

Remark 16 (Spelling variants and arithmetic coding)**.** *Given a word embedding (the σ vector concentrated on "the"), how much additional information is needed to recover the exact spelling? For standard spelling, zero bits—the word determines its bytes. For a variant spelling ("teh"), we need to encode the deviation from canonical.*

The luck $\lambda = 1/P(variant)$ gives the cost in bits: for "teh" occurring with probability $P = 0.001$ among "the" tokens, the extra cost is $\log_2(1/0.001) \approx 10$ bits. This matches arithmetic coding: the embedding carries the word identity, and the residual (spelling variant) is encoded by its luck.

More generally: the $E \to N \to Q$ chain handles the marginalization from bytes to words and back. The Bayesian conditioning (partial quotient [?]) lets us go from the word embedding (coarse) to the byte sequence (fine) with a cost equal to the within-word entropy—the information that the word-level representation "externalizes."

10 Why This Differs from the Experiments

Proposition 17 (Mixing $\neq$ extending)**.** *The* `the_inject` *experiments mixed predictions:*

$$P_{mix}(o) = \alpha\, P_{trie}(o) + (1 - \alpha)\, P_{RNN}(o). \tag{11}$$

The extended event space computes predictions from patterns over the full $I' \times H' \times O'$:

$$P_{extended}(o) = f_{p'}(t')_o = \max_{i' \in I'} \min(t'_{i'}, p'_{i',o}), \tag{12}$$

where t' is the extended support vector (including position, accumulator, and word events) and p' is the extended pattern table (including word-to-byte patterns).

The difference:

	Mixing	*Extending*
Shared state?	*No*	*Yes (shared H')*
New patterns?	*No*	*Yes (word→byte, word→word)*
New events?	*No*	*Yes (position, accumulator, words)*
Trainable?	*α only*	*Full ω_0 over extended E*
UM-native?	*No (ensemble)*	*Yes (single UM)*

The critical difference is *shared representation*. In the extended UM, the word event "we are in 'the' " is part of H', visible to every pattern that reads H'. The word-to-byte pattern "after 'the', space is likely" coexists with the byte-to-byte pattern "after 'e', many continuations are possible." The forward pass $f_{p'}$ resolves them via max-min: the word event provides stronger support for "space" than the byte event provides for alternatives, so the word event wins.

In the mixing approach, the trie and RNN never see each other's state. The RNN doesn't know the trie thinks we're in "the"; the trie doesn't know the RNN's hidden state suggests XML context. The only coordination is the global α.

11 The Research Agenda

1. **Neutral injection into the extended UM.** Build the extended event space $I' \times H' \times O'$ with "the" as the first word event. Verify that marginalizing over E_{the} recovers the base model's predictions exactly (as the neutral factorization guarantees).

2. **Add intra-lexical patterns.** Compute the patterns from the extended input (position, accumulator) to the word event, and from the word event to the output. Measure Δbpc from these patterns alone.

3. **Add "and" as the second word.** Inject the second word event. Measure the individual and interaction ΔI. Does "and" synergize with "the" (positive interaction) or is it redundant (negative)?

4. **Iterate through the vocabulary.** Inject words in frequency order, measuring the injection curve [**?**]. Track the embedding $\sigma(t)$ as it grows.

5. **Visualize the product pattern.** The LPP (log product pattern) between two event spaces can be visualized as two rings (one per ES) with nodes distributed by probability along each circumference, fully connected with connection strength encoded as thickness or opacity. Build interactive visualizations for the word-to-byte and word-to-word patterns.

6. **Compare against KN baseline.** The extended UM must be compared against KN-5 at 10M (2.39 bpc split) and KN+RNN (2.33 bpc). The goal is not to beat KN by mixing, but to build an interpretable architecture that achieves competitive bpc via named events and counted patterns.

7. **Scale toward Hutter Prize.** The state of the art is $\sim$1 bpc. The extended UM, with a full lexicon and word-to-word patterns on top of byte-level KN, should approach this target via a purely counting-based, interpretable, gradient-free architecture.

12 Discussion

12.1 The tock as architecture extension

The tock step [**?**] discovers new event spaces from data. This paper makes the tock concrete for lexical structure: the new events are in-word position, letter accumulator, bag-of-letters, graded word support, and word identity. Each is computable from the byte stream via counting (ω_0) or deterministic logic (strength 255 patterns).

The key principle: *extending H is not the same as mixing predictors.* Extending H adds new events that participate in patterns; mixing predictors combines black-box outputs. The UM framework demands the former.

12.2 The RNN as proof of concept

The RNN proved that rotational embeddings work [**?**]: the shift-register dynamics of W_h create a carrier signal that encodes byte information at multiple offsets. The factor map proved that every neuron computes a 2-offset conjunction. The weight construction [**?**] closed the loop by deriving all 82k parameters from data statistics.

Now we cut to the capability directly. The in-word position event replaces the carrier signal. The bag-of-letters replaces the conjunction detectors. The word-level H' replaces the implicit word

encoding in the hidden state. Everything the RNN does is expressible in the extended event space—plus things the RNN cannot do (explicit word-to-word patterns, symmetric I/O, arbitrary offset connections).

12.3 What we expect to find

The neutral injection will be trivially exact (it was already proved). The first non-trivial result will be the Δbpc from intra-lexical patterns for "the"—the value of knowing the word identity at each byte within the word. The experiment paper [**?**] measured this with oracle labels (0.130 bpc for "the") and causal prefix trie (4.57 bpc at 500 words). The extended UM should achieve similar gains via event-space patterns rather than external mixing.

The real test is scaling: does the extended UM's injection curve converge to competitive bpc with the right shape (concave, with function words in the high-gain regime and content words in the linear regime)? If yes, the tock has produced a domain-native architecture for English text—interpretable, gradient-free, and competitive.

References

[1] Michaeljohn Clement. *CMP*. `https://cmpr.ai/cmp.pdf`, 2026.

[2] Claude and MJC. *The Tock Protocol: Systematic Lexicon Injection into the Isomorphic UM*. Hutter archive, 13 Feb 2026.

[3] Claude and MJC. *The Tock Step: Domain-Native Architecture from Evidence*. Hutter archive, 12 Feb 2026.

[4] Claude and MJC. *Lexemes as Binary Event Spaces: From Atomic Patterns to Bag-of-Letters Prediction*. Hutter archive, 14 Feb 2026.

[5] Claude and MJC. *Introducing "the": A Neutral Factorization of the Byte-Level Model*. Hutter archive, 14 Feb 2026.

[6] Claude and MJC. *Lexeme Injection Experiments: From Neutral Factorization to KN Dominance*. Hutter archive, 14 Feb 2026.

[7] Claude and MJC. *The Real Factor Map: Interpretable Patterns onto Hidden Dynamics*. Hutter archive, 9 Feb 2026.

[8] Claude and MJC. *The Carrier Signal Problem: Why Product Patterns Need Orthogonal Offsets*. Hutter archive, 12 Feb 2026.

[9] Claude and MJC. *Weight Construction: All 82k Parameters from Data Statistics*. Hutter archive, 11 Feb 2026.

[10] Claude and MJC. *Bayes from Counting: Partial Quotients, GCD, and the Symmetric Learning Function on $E = I \times O$*. Hutter archive, 12 Feb 2026.