Context Events and the Induction of Priors

v3: Generalized Oversupport

Claude and MJC

18 February 2026

Abstract

We formalize the concept of a *context event*: an atomic event in a causally prior event space that, through the UM forward pass, induces a prior distribution on downstream event spaces. We conjecture that every missing prior in P-programming corresponds to a missing context event. Central to this version: we generalize oversupport beyond predicted-vs-observed conflict to any ES with strong support for multiple events—which contradicts the ES epistemics that events are mutually exclusive. This can arise from sensory conflict, from multiple LPPs terminating at the same ES ("from the left"), or from internal model inconsistency. The organism's response is the same in all cases: immediate energetic expenditure to resolve the contradiction.

1 Introduction

In the UM framework, compression proceeds by learning joint event distributions across event spaces via Learned Probabilistic Patterns (LPPs). The KN-6 model achieves 1.784 bpc on enwik9 using a tower of byte-context event spaces connected by LPPs with KN smoothing.

Yet significant compression remains on the table. The gap between KN-6 (1.784 bpc) and the current best compressed size arises because KN smoothing uses a crude sliding window as context, discarding most sequential information.

We argue that this gap is precisely characterized by *missing context events*: atomic events that, if present in the model, would condition the downstream LPPs to produce tighter output distributions.

2 Definitions

Definition 1 (Event Space)**.** *An event space* ES_a *is a named set of atomic events* $\{e_1, \ldots, e_n\}$ *with a support function* $s : \mathrm{ES}_a \to [0, 255]$ *assigning each event a current support value. The ES epistemics: events within an ES are* mutually exclusive*. At most one event in the ES should have high support at any time.*

Definition 2 (Context Event)**.** *Given a target LPP from* ES_a *to* ES_b*, a* context event $e_c \in \mathrm{ES}_c$ *is an atomic event in a causally prior event space such that the LPP, conditioned on* e_c *having nonzero support, produces a more accurate output distribution on* ES_b*.*

Definition 3 (Prior Induction)**.** *When a context event* e_c *has support* $s(e_c) = k > 0$*, and* e_c *is connected to events in* ES_a *via atomic patterns with weights* $w_{c,i}$*, the UM forward pass induces a prior on* ES_a*:*

$$s(e_i) \leftarrow \max\big(s(e_i),\ \min(k, w_{c,i})\big)$$

This is the standard max-min update. The pattern of weights $\{w_{c,i}\}$ *characterizes which events in* ES_a *are "expected" under context* e_c*, and with what strength.*

3 Generalized Oversupport

3.1 Beyond Predicted vs. Observed

In v2, oversupport was defined as a conflict between the model's prediction and an external observation: the model supports e but the world forces $x \neq e$. This is correct in the "inner UM" sense—the output ES under teacher forcing—but the more general statement is:

Definition 4 (Oversupport (Generalized))**.** *An event space* ES_a *exhibits* oversupport *when two or more events in* ES_a *have strong support simultaneously:*

$$|\{e \in \mathrm{ES}_a : s(e) > \tau\}| \geq 2$$

This contradicts the ES epistemics: events in the same ES are mutually exclusive, so strong support for multiple events is a contradiction.

Definition 5 (Oversupport Magnitude)**.** *The magnitude of oversupport at* ES_a *is the* second-highest support *in the ES:*

$$\mathit{oversupport}(\mathrm{ES}_a) = \mathit{sort}(\{s(e) : e \in \mathrm{ES}_a\})_2$$

interpreted relative to the total incoming energy flux—the total support for the whole ES, which should equal the support for some context event that projects onto this ES.

If the second-highest support is zero, there is no oversupport: only one event is supported, consistent with the ES epistemics. If the second-highest support is high, the ES is in contradiction.

3.2 Sources of Oversupport

Oversupport can arise from multiple sources:

1. **Sensory conflict**: an external observation forces one event, but the model's forward pass supports another. This is the teacher-forcing case (output ES). But even sensory input isn't certain—the organism may expend energy (e.g., a double-take) to verify a percept. Typos in the data are an example even under teacher forcing.
2. **"From the left"**: multiple LPPs or pattern chains terminating at the same ES support different events. No observation is involved—the contradiction is entirely internal to the model. Different upstream contexts lead to conflicting conclusions at this ES.
3. **Model inconsistency**: the ES's events are not actually mutually exclusive for the data, meaning the ES factorization is wrong. The "events" are not really events in the UM sense.

3.3 What Oversupport Means

We emphasize: oversupport at an ES does *not* necessarily mean "the model is confidently wrong." It has three possible interpretations:

1. The model *is* wrong: the support pattern reflects an incorrect hypothesis. This is the "model error" case.
2. The ES *isn't really an ES*: the events aren't truly mutually exclusive. The factorization of E is wrong—what looks like one ES should be two, or the events need splitting.

3. Something *genuinely rare* happened: it's raining but the grass isn't wet, because there's a zeppelin in the garden. The model is correct about the general case; this is just an unusual instance.

The organism's response is the same in all three cases: immediate energetic expenditure to figure out which explanation applies. This is the startle response—"freeze and figure out what's going on."

3.4 The Binary Case: Argument by Contradiction

In a binary event space ES = $\{e_0, e_1\}$:

- **Undersupport = excluded middle**: neither event has support. The model has nothing to say. Violates the law of the excluded middle.
- **Oversupport = contradiction**: both events have support. This is a logical contradiction—both True and False are asserted.

The generalization matters here: oversupport in the binary case can arise "from the left" without any observation. If one LPP supports e_0 and another supports e_1, that is a contradiction *within the model*—an argument by contradiction. The model's own premises lead to incompatible conclusions.

This internal contradiction is a signal that something is wrong with the premises (the upstream patterns), not necessarily with the observation.

Both cases generalize naturally to non-binary ESs.

3.5 Prediction Loss as a Special Case

The identification of prediction loss with oversupport (from v2) is a special case of the generalized definition. At the output ES under teacher forcing:

- The model supports e (via the forward pass).
- The observation forces $x \neq e$ (via the input).
- Both e and x have high support: oversupport.

The cross-entropy loss $\mathcal{L}_t = -\log P(x_t)$ measures this oversupport. Backpropagation computes $\nabla_\theta \mathcal{L}_t$—the gradient of the oversupport with respect to model parameters. But this is just one instance of oversupport (the sensory-conflict case at the output ES). The generalized definition captures internal contradictions that backpropagation cannot see, because they occur at ESs with no explicit observation.

Remark 1 (The Ramification Principle)**.** *Just as the special treatment of the output ES (softmax, loss, gradient) is a signal to ramify the mechanism throughout, the detection of oversupport should not be limited to ESs with external observations.* Every *ES can exhibit oversupport (generalized), and the detection mechanism—measuring the second-highest support—requires no external observation at all.*

4 The Conjecture

Conjecture 1 (Missing Prior = Missing Context Event)**.** *In P-programming, whenever a model exhibits systematic oversupport at any ES—a pattern of contradiction that repeats—there exists a context event e_c in some ES such that adding e_c and its connections to the model eliminates the systematic component.*

This refines the v2 conjecture: it's not just systematic prediction errors (output-ES oversupport) but systematic contradictions at *any* ES that indicate a missing context event.

5 KN Smoothing as Crude Context

In KN-6, the context for predicting byte t is the preceding k bytes $(b_{t-k}, \ldots, b_{t-1})$. The KN interpolation chain (order $6 \to 5 \to \cdots \to 1 \to$ unigram) is a tower of progressively coarser contexts.

What KN gets right: the sliding window captures local sequential dependencies.

What KN misses: everything outside the window. The context event for "we are inside an HTML tag" requires information from potentially hundreds of bytes earlier. KN-6 cannot represent this.

Our surprise analysis confirms this empirically: KN-6 achieves 0.38 bpc inside XML tags but 2.46 bpc outside them (86.8% of mean surprise). The tag structure is highly predictable, but only if you know you're inside a tag—a context event that the 6-byte window almost never captures.

6 Neural Reuse via Context Events

A central problem in compression: the same local patterns (e.g., "th" $\to$ "e") occur in many different contexts (normal text, tags, attributes, URLs). A single LPP captures the average, losing the context-specific distribution.

Context events solve this: the LPP between byte context and byte output remains the same, but different context events in a prior ES activate different subsets of the LPP's patterns.

Example 1. *Consider the bigram LPP from 1-byte context to output. In KN-6, $P(\text{'e'} \mid \text{'h'})$ is a single number. But with context events:*

- $e_{\text{word\_start}}$ *active: "h" is likely "he", "his", "had"*
- $e_{\text{in\_tag}}$ *active: "h" is likely "href", "html"*
- $e_{\text{after\_digit}}$ *active: "h" is likely a hex digit*

Each context event induces a different prior over the output ES, restricting the LPP to its relevant joint events. The oversupport that KN-6 experiences when predicting tag-interior "h" (expecting text patterns) is eliminated.

7 The Boolean Interpretation

Every context event has a Boolean reading. The max-min forward pass is:

$$s(y) = \max_{\text{patterns}} \min(s(x_1), s(x_2), \ldots, s(x_n), w)$$

which, when all supports are binary ($\{0, 255\}$), reduces to DNF:

$$y = \bigvee_{\text{patterns}} (x_1 \wedge x_2 \wedge \cdots \wedge x_n)$$

A context event adds one more term to each conjunction:

$$y = \bigvee (c \wedge x_1 \wedge x_2 \wedge \cdots \wedge x_n)$$

where c is the context. This is the probabilistic syllogism.

7.1 Finding Exceptions via Boolean Query

In the Boolean interpretation, oversupport at an ES reveals *exceptions to the currently represented rule.* Finding the corrective context event c can be seen as a query over the dataset, looking for exceptional cases.

Example 2. *If 'h' appears after* $e_{\text{in\_tag}}$ *and this combination is rare, we have a set of exceptions to the default rule for 'h'. These exceptions are expressible in natural language as a probabilistic syllogism (when the events are labeled), and literally correspond (via* $E \leftrightarrow \mathbb{N} \leftrightarrow \mathbb{Q}$*) to a subset of the dataset that activates that Boolean circuit.*

An LPP over the potential-context ES and the "oversupport event" (the second-most-supported event at the target ES) can be a way to find these exceptions. This is suggestive of an "embedding into P" implementation of surprise detection: the model itself, through its own LPPs, discovers which context events would resolve which oversupport events.

This remains a research direction. The ring pattern approach (see the surprise-mechanism paper) is one possible P-programmish implementation.

8 The Methodology

We propose a systematic procedure for identifying and implementing context events:

1. **Identify systematic oversupport.** Score the data with the current model. Find positions where oversupport repeats in a pattern—not isolated high-surprise events, but clusters sharing a common feature. Generalized oversupport (second-highest support at any ES, not just output) gives richer signal.

2. **Boolean interpretation.** Express the pattern: "the model systematically oversupports e when the actual situation involves x, whenever [some condition holds]."

3. **Natural language.** State the condition: "we are inside an HTML tag", "the current word is a number", "we just saw a capital letter after a period."

4. **Reify via E/N/Q.** Express the condition as an event in the integer representation (um-arithmetic-v4): identify the quotient space, the projection, the relevant equivalence class.

5. **Query the dataset.** Find concrete examples. Verify that the condition separates high-oversupport from low-oversupport positions. This query *is* the Boolean circuit from step 2, applied to the data.

6. **P-program.** Implement the context event as an ES with events detected by a P-program. Connect it to the target LPP's input ES. The model can now condition on the context, eliminating the systematic oversupport.

9 Empirical Validation

Our surprise analysis (`umr surprise enwik9 1000000`) provides the first empirical test of the context events framework:

- **in_tag**: 0.38 bpc inside tags vs. 2.46 bpc outside (86.8% of mean). This is the single most powerful context event we have identified. It would eliminate the systematic oversupport that KN-6 experiences at tag boundaries.

- **word_len**: Boundaries (word_len=0) have 3.98 bpc; mid-word (len 3–4) has 1.73 bpc. Word position is the second most powerful context event.

- **Surprise distribution**: 40.7% of positions have < 0.25 bpc (nearly free); 8% have > 7.75 bpc (nearly random). The heavy tail is where context events are most needed.
- **Top surprises**: 44–54 bits per byte at positions with rare markup, non-ASCII characters, and structural transitions. These are extreme oversupport events.

10 Open Issues for P-Programming

10.1 Ramifying Oversupport Detection

The key open problem: how to give every ES in the model oversupport detection. With the generalized definition, this is simpler than v2 proposed—we need only measure the second-highest support at each ES, which requires no external observation. The question is what the model *does* with this signal.

10.2 Discovery

How do we systematically discover which context events to add? The Boolean query approach (find the dataset positions that activate the oversupport circuit, look for shared features) is promising. An LPP between potential-context ESs and oversupport signals could automate this—making the model self-improving.

10.3 The Carrier Signal

Memory traces can be stored "inline" via a carrier signal: an event representing a span of nonzero duration λ. The joint event $(e., t_i, e., t_{i+1})$ encodes temporal adjacency and can be recorded by an LPP. This converts sequential memory into associative memory. We defer detailed treatment to a future paper.

11 Connection to the KN-Quotient Tower

The kn-quotient papers (20260216 archive) established that KN smoothing has an algebraic structure: contexts are elements of $\mathbb{Z}/256^k\mathbb{Z}$, order reduction is a ring surjection, and the discount D corresponds approximately to removing common evidence via GCD.

Context events extend this tower upward. The KN sliding window is *already* a context event mechanism—just a crude one. What we propose is a generalization: richer context events, detected by P-programs rather than fixed sliding windows, with oversupport detection at every level of the tower rather than only at the output.

12 Conclusion

The context event concept, combined with generalized oversupport, unifies several threads:

- Oversupport is strong support for multiple events in the same ES—a contradiction of the ES epistemics. It can arise from sensory conflict, from internal disagreement ("from the left"), or from a wrong factorization of E. All three cases demand the same organism response: immediate energetic expenditure to resolve.
- Cross-entropy loss at the output ES is a special case (sensory oversupport under teacher forcing). The general mechanism detects contradictions at any ES, including internal ones invisible to backpropagation.

- In a binary ES, undersupport is the excluded middle, oversupport is contradiction. Internal oversupport from the left is argument by contradiction.
- Finding corrective context events is a Boolean query over the dataset: which positions activate the oversupport circuit? An LPP between context ES and oversupport signal can automate this discovery.
- The methodology (identify systematic oversupport → Boolean → NL → E/N/Q → query → P-program) provides a path from contradiction to model improvement that stays within the UM paradigm.

The next step is to implement generalized oversupport detection in the UM runner (second-highest support at each ES) and use it alongside the Boolean query mechanism to discover context events automatically.