The Timing Resolution

Solving Marginal Dominance via Pattern Chain Length

Claude and MJC — February 19, 2026

1. The "the" Problem

Consider a UM bigram model with event spaces E_I (byte input), E_O (byte output), E_{env} (envelope), and E_B (bigram). The forward pass is $f_p(t)_j = \max_i \min(t_i, p_{ij})$.

After observing the sequence "th", the support vector contains $t_{\text{“h”}} = 255$ (input) and $t_{\text{“th”}} > 0$ (bigram). Two paths project onto the output ES:

$$\begin{aligned} \text{Marginal path:} \quad & \text{input[“h”]} \to \text{output[“e”]} \quad \text{weight } w_1 = \log_2 \text{count(“he”)} & (1) \\ \text{Bigram path:} \quad & \text{bigram[“th”]} \to \text{output[“e”]} \quad \text{weight } w_2 = \log_2 \text{count(“the”)} & (2) \end{aligned}$$

(Weights are stored as integers via the log-stochastic increment; the stochastic rounding error is negligible here.)

Since every occurrence of "the" contains "he" but not vice versa, $w_1 \geq w_2$ always. Under max-min, the marginal wins:

$$\text{support[output[“e”]]} = \max(\min(255, w_1),\ \min(s_{\text{th}}, w_2)) = w_1$$

The bigram signal is invisible. This is **marginal dominance**: lower-order patterns with more observations always dominate higher-order patterns under max-min with absolute log-counts.

The fundamental issue is that event spaces with different total support cannot be mixed. The "h" event fires with higher support than "th", so when both project onto the output ES, the marginal always wins—even though the conditional $P(e \mid th)$ is much higher than $P(e \mid h)$. This is analogous to comparing Bayesian evidence (log-odds ratios) with different denominators.

2. Why Obvious Solutions Fail

The Bayesian answer is clear: marginalize, or subtract the "h" contribution when "th" is present (since "th" fully explains "h"). But:

- **Subtraction** is non-physical—not only in the UM but in the brain. No biological neuron subtracts evidence from a competing hypothesis. The forward pass has only max and min: no subtraction, no negative weights.
- **Inhibition** (a pattern that reduces support) would break the algebraic closure. Patterns only increase support; they are evidence *for*, never *against*.
- **Normalization** (dividing each ES's output by its total support) introduces a non-local operation—each event must know the global sum across its ES.

Any ad-hoc extension to handle this one case would compromise the UM's generality.

3. The Timing Resolution

The key observation: the bigram pattern necessarily has a **longer causal chain** than the marginal pattern.

Path	**Chain length**	**Sharpness**
input[“h”] → output	1 pattern	Broad $(P(\cdot \mid h))$
input[“h”] → bigram[“th”] → output	2 patterns	Sharp $(P(\cdot \mid th))$

These paths have different lengths and therefore introduce a **timing disparity** within a single application of f. We call this a *triangle pattern*: two chains with the same endpoints but different lengths. The short chain (marginal) and the long chain (bigram) both terminate at the output ES, but the long chain carries more context.

3.1. Why f Represents Timing

The timing disparity within a single f is not an implementation detail of LPP ordering—it is a fundamental property. f must represent timing information for two reasons:

1. **Brain simulation.** A realistic model of neural computation must encode the fact that signals propagate through different path lengths at different speeds. In the brain, not every synapse has the same length; there is enormous flexibility in the timing of signal arrival. Triangle patterns are the minimal abstract representation of this physical reality.
2. **Capability.** We find that we *need* timing-based resolution for capabilities (such as using higher-order context) that the UM cannot otherwise express. The timing structure is not decorative—it is load-bearing.

The derivation of layers from the pattern graph P (see *Deriving Layers from the Connectome*, Feb 18) provides the timing structure naturally. Patterns that form chains through intermediate ESs define a depth ordering; this depth IS the timing. No additional mechanism is needed to “create” timing—it emerges from the pattern topology.

In silico, we may need to explicitly introduce orderings or “slow down” certain paths to recreate the timing effects that Nature gets for free from physical synapse lengths. This is a rich source of P-programming techniques: the biological timing flexibility maps onto a large space of pattern topologies we can explore.

3.2. The Resolution Principle

The timing disparity distinguishes pattern chains by length: they arrive at different times. This provides a plausible mechanism for **sharpness preference** to operate, because in a brain, neurons fire at some frequency—not like setting a continuous voltage on a line. The discrete arrival of signals at different times makes the contributions of different chains locally observable.

The longer chain that arrives later is *not necessarily better.* It has the *potential* to be better, because it accesses richer context. But the preference signal is sharpness itself, aligned with ES epistemics

(Section 4). Timing merely provides a way for local observation to distinguish between patterns that would otherwise be indistinguishable—because pure accumulation of support from patterns of all lengths destroys the distinction entirely.

This separation is critical: timing is the *observation mechanism*; sharpness is the *preference signal.* Without timing, the ES sees only the max-min aggregate, which is dominated by the marginal. With timing, each chain's contribution can be evaluated independently, and the sharper one preferred.

Crucially, there is no requirement that every input-output cycle takes the same time. The **input-dependence of the output timing** is itself a strong signal that this resolution is correct: harder inputs (less context available) resolve faster to broad predictions; easier inputs (rich context, long chains) take longer to settle on sharp predictions. Varying response time is not a bug—it is information about the model's confidence.

This works because:

1. **Longer chains have the potential for sharper distributions.** A length-k chain carries k events of context. Empirically, longer contexts produce sharper predictions (KN-6 beats KN-1). But the longer chain is not *automatically* sharper—it is sharper when the additional context is informative.
2. **Sharpness is already measurable.** The ring pattern construction (see *The Ring Pattern: Measuring Oversupport in the Universal Model*, Feb 18) provides an existing UM mechanism for measuring how concentrated support is within an ES. The second-highest support $s^{(2)}$ measures oversupport; its complement measures sharpness. No new machinery is needed—the ring pattern IS the sharpness detector.
3. **Causal attribution is singular.** The "th" event fully explains the "h" event (every "th" contains an "h"). Counting both is double-counting. The longer explanation subsumes the shorter, and the timing mechanism makes this subsumption observable.
4. **The mechanism is physical.** In the brain, neurons fire at frequencies; signals propagate through chains of synapses with different physical lengths. The timing disparity between short and long chains is not an abstraction—it is a physical reality that the organism exploits. No subtraction, no inhibition, no normalization. Just local observation of arrival times and preference for the sharper signal.

3.3. Generality

This is a general principle, not special to the output ES. It applies to *any* ES in the UM that receives signals from chains of different lengths. For any such ES, timing makes the per-chain contributions distinguishable, and the sharpest chain's signal is preferred—regardless of whether it is the longest or shortest. This will be a basic P-programming technique across all future models.

4. The One-Back P-Program

The bigram model requires knowing the previous input byte. In our current implementation, this is *cheated*: the C code maintains a variable `prev` and directly injects bigram events into the support vector, bypassing f entirely. The bigram neuron “th” never fires from the model—it is conjured by the caller.

This violates the fundamental principle: **if the capability doesn’t live in the model, we haven’t proved anything.** A UM result must be a UM result. Experiments construct models; the generic runner scores them. The runner’s job is to provide sensory input (the current byte, the envelope) and read the output. Everything else is the model’s responsibility.

The proper architecture:

1. A **prev_byte** ES stores the previous input byte as part of the sensory boundary. (The runner sets this, just as it sets the current byte—it is sensory memory, not internal computation.)
2. **Conjunction patterns** activate bigram events. A conjunction pattern has two source events: $(a, b) \to c$ with weight w, computing $\min(t_a, t_b, w)$ and maxing into t_c. This extends the atomic pattern from single-source to multi-source.
3. The bigram LPP projects from bigram events to output. Because this goes through bigram[“th”], it is a length-2 chain and arrives *later* than the direct input→output marginal.

The conjunction extension is minimal: each pattern entry gains an optional second source. The forward pass becomes:

$$\text{val} = \min(t_{\text{from}_1},\ t_{\text{from}_2},\ w) \qquad (\text{where } t_{\text{from}_2} = 255 \text{ if single-source})$$

4.1. Future: The Ring Buffer

Providing prev_byte as sensory input is the honest interim solution. The full solution is a P-program that maintains a ring buffer inside the UM, with a pointer-like timing signal to track the current write position. This is far more efficient than 256 identity patterns copying onto a second byte ES (which has the accumulation problem under max-only updates). The ring buffer is the first P-programming technique we need to discover and document.

5. Architectural Principles

From this analysis, three principles for the UM research program:

1. UM results must be UM results. If the runner hacks a feature in, it’s not a UM capability. The runner is generic: it provides sensory input, runs f, reads output. All computation lives in the model (ESs, LPPs, patterns).

2. Experiments construct models, not runners. The experiment’s job is to build an SN file. The runner’s job is to score it. There should be no experiment-specific code in the runner. `#umr_bigram` is wrong—it should be a model construction tool that outputs an SN file, scored by the generic `umr run`.

3. The forward pass provides timing for free. Pattern chain length within a single f is a timing signal. Triangle patterns—two chains with the same endpoints but different lengths—are the minimal unit of timing disparity. Timing makes per-chain contributions distinguishable; sharpness preference selects among them. This is a P-programming primitive, and the biological analogy (variable synapse length, frequency-coded firing) suggests a rich space of timing-based techniques to explore.

6. Implementation Status

Done	Forward pass, SN format, generic runner, JS viewer
This paper	Multi-source conjunction patterns in `#umr_core`
	Model construction tool (replaces `#umr_bigram`)
	Proper bigram model as SN file, scored by generic runner
Next	Settling mechanism (ring pattern as sharpness detector)
	Ring buffer P-program for internal one-back storage
	Scale to KN-6 parity via P-programming