Appendix B: M/L/D Algorithm Specifications

For: *Memory Without Storage, Learning Without a Learner* (Neale, 2026)

B.1 Overview

All three measures derive from a single rolling predictive model applied to the emission stream E(t) ∈ ℝ^d, where d is the number of emission features and t = 1, ..., T. The observer has no access to the system's internal dynamics — only the emission stream.

The measures are:

- **M (Memory):** Model obsolescence — an early-trained predictor fails on late data
- **L (Learning):** Bayesian surprise reduction — the observer's predictions improve over time
- **D (Decision):** Surprise spikes — the observer's predictions suddenly fail

All three are computed from the same surprise signal, making them aspects of a single predictive process rather than independent measurements.

B.2 The Observer's Predictive Model

The observer maintains a rolling mean predictor with window size w:

```
FUNCTION observer_predict(E, window):
    T, d ← shape(E)
    w ← min(window, max(2, T / 10))          # adaptive window

    FOR t = w TO T-1:
        # History-based prediction: rolling mean of recent emissions
        prediction_history ← mean(E[t-w : t], axis=features)
        surprise_history[t] ← RMSE(E[t], prediction_history)

        # Present-only prediction: just the previous emission
        prediction_present ← E[t-1]
        surprise_present[t] ← RMSE(E[t], prediction_present)

    RETURN surprise_history, surprise_present
```

where RMSE is root mean squared error across emission features:

$$\mathrm{RMSE}(\mathbf{a}, \mathbf{b}) = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (a_i - b_i)^2}$$

The window size w is set to 10 for most substrates and 5 for sorting algorithms (which have fewer timesteps). It is capped at max(2, T/10) to prevent the window from exceeding a tenth of the total run length.

B.3 Memory (M): Model Obsolescence

Concept: The observer builds a predictor from early emissions and applies it to late emissions. If the early model fails on late data worse than a predictor built from late data, the system has moved persistently — the observer infers "memory."

Primary measure (M_pred):

```
FUNCTION measure_M_pred(E, window):
    T, d ← shape(E)
    third ← T / 3
    IF third < 10: RETURN 0

    early ← E[5 : 5 + third]           # skip first 5 steps
    late  ← E[T - third : T]
    w ← min(window, third / 3)

    # Early model: static mean of early emissions
    early_mean ← mean(early, axis=time)

    # Error of early model applied to late data
    error_early_on_late ← mean over t in late:
        RMSE(late[t], early_mean)

    # Error of late rolling model applied to late data
    error_late_on_late ← mean over t = w to len(late):
        RMSE(late[t], mean(late[t-w : t], axis=time))

    # Memory = how much worse the stale model is
    M ← max(0, error_early_on_late - error_late_on_late)

    RETURN M
```

Interpretation:

- M = 0: Early model still works on late data. No persistent change. No memory inferred.
- M > 0: Early model fails on late data. The system moved to a new emission regime. Memory inferred.
- Large M: The regime shift was dramatic. Strong memory.

Validation measure (M_cohen):

As an independent check, we also compute Cohen's d between early and late emission distributions:

```
FUNCTION measure_M_cohen(E):
    T, d ← shape(E)
    third ← T / 3

    early ← E[5 : 5 + third]
    late  ← E[T - third : T]

    FOR each feature f = 1 to d:
        mean_diff ← |mean(late[:, f]) - mean(early[:, f])|
        pooled_std ← sqrt(((n1-1)*var(early[:,f]) + (n2-1)*var(late[:,f])) /
(n1+n2-2))

        IF pooled_std > ε:
            cohen_d[f] ← mean_diff / pooled_std
        ELSE IF mean_diff > ε:
            cohen_d[f] ← 10.0           # large shift from constant to constant
        ELSE:
            cohen_d[f] ← 0.0            # both distributions identical

    M_cohen ← mean(cohen_d)
    RETURN M_cohen
```

M_pred and M_cohen are reported together. Their Spearman correlation across systems validates that both capture the same underlying phenomenon.

B.4 Learning (L): Bayesian Surprise Reduction

Concept: The observer's predictions improve over time while the system remains active. Surprise decreases from the early phase to the late phase. The system becomes less surprising — the observer infers "learning."

```
FUNCTION measure_L(E, window):
    T, d ← shape(E)

    # Compute surprise signal
    surprise ← observer_predict(E, window).surprise_history

    # Identify valid timesteps (where surprise is computable)
    valid ← indices where surprise is not NaN
    IF len(valid) < 10: RETURN 0

    # Divide into thirds
    third ← max(3, len(valid) / 3)
    early_indices ← valid[0 : third]
    late_indices  ← valid[-third : end]

    # --- Test 1: Did surprise decrease? ---
    s_early ← mean(surprise[early_indices])
    s_late  ← mean(surprise[late_indices])

    IF s_early < ε:
        reduction ← 0                  # was never surprising
    ELSE:
        reduction ← (s_early - s_late) / s_early

    # --- Test 2: Is the system still active? ---
    # Compute mean absolute emission change in early and late phases
    early_activity ← mean(|diff(E[early_indices], axis=time)|)
    late_activity  ← mean(|diff(E[late_indices], axis=time)|)

    activity_floor ← 0.01 × early_activity
    system_alive ← late_activity > activity_floor

    # --- Combine ---
    IF reduction > 0 AND system_alive:
        L ← reduction
    ELSE:
        L ← 0

    RETURN L
```

The activity gate prevents dead systems from registering as "learning." A system that ceases all activity (e.g., CA Class 1 reaching uniform state, or Diehard completing its die-off) produces zero late surprise because the emissions become constant — but the observer should not infer learning from cessation. The gate requires that late-phase emission variability exceeds 1% of early-phase variability.

Robustness: The 1% threshold is not a sensitive parameter. Sensitivity testing across all 256 CA rules at thresholds ranging from 0.1% to 5% (a 50-fold range) produced zero changes in $L > 0$ classification. The gate is effectively binary in practice: systems are either clearly alive (late activity well above any reasonable floor) or clearly dead (late activity is zero or negligible). The only system in the full test set that is sensitive to the threshold is the GoL R-pentomino, whose late-phase activity (residual oscillators producing ~2% of early-phase variability) falls below the gate only at thresholds above 2%.

Interpretation:

- $L = 0$: Either surprise didn't decrease, or the system died. No learning inferred.
- $L > 0$: Surprise decreased while the system remained active. Learning inferred.
- $L = 0.99$: Surprise dropped 99%. The system became almost perfectly predictable while still producing variable emissions. Strong learning (e.g., R-pentomino resolving from chaos into oscillators).

Connection to Friston: Surprise $s(t)$ is the empirical proxy for the negative log-evidence in variational inference. Decreasing surprise is decreasing free energy: $L > 0 \Leftrightarrow \Delta F < 0$.

B.5 Decision (D): Surprise Spikes

Concept: The observer's prediction suddenly fails. The current emission is far more surprising than recent emissions. The observer infers "a decision" — the system did something the observer did not expect.

```
FUNCTION measure_D(E, window, threshold):
    T, d ← shape(E)

    # Compute surprise signal
    surprise ← observer_predict(E, window).surprise_history
    valid ← indices where surprise is not NaN

    IF len(valid) < window + 5: RETURN 0, 0, []

    s_vals ← surprise[valid]
    d_window ← min(15, len(valid) / 4)

    decision_events ← []
    decision_times  ← []

    FOR i = d_window TO len(s_vals) - 1:
        recent ← s_vals[i - d_window : i]

        # Robust baseline: median
        median ← median(recent)

        # Robust scale: median absolute deviation
        mad ← median(|recent - median|)

        # Compute z-score (MAD-based)
        IF mad > ε:
            z ← (s_vals[i] - median) / mad
        ELSE IF median > ε:
            z ← s_vals[i] / median       # fallback for constant baseline
        ELSE:
            z ← 0                         # both baseline and current are ~0

        IF z > threshold:
            decision_events.append(z)
            decision_times.append(valid[i])

    # Decision rate: fraction of eligible timesteps with decision events
    n_eligible ← len(s_vals) - d_window
    D_rate ← len(decision_events) / n_eligible

    # Decision magnitude: median surprise of decision events (in MAD units)
    D_mag ← median(decision_events) IF len > 0 ELSE 0
    RETURN D_rate, D_mag, decision_times
```

Why MAD instead of standard deviation: The median absolute deviation is robust to the very outliers we are trying to detect. Standard deviation is inflated by surprise spikes, raising the threshold and masking subsequent events. MAD-based detection is self-consistent: outliers do not inflate the baseline.

Threshold: Set at 2.0 MAD units throughout. This is analogous to a z > 2 criterion but robust. A timestep registers as a "decision" if its surprise exceeds the local median by more than twice the local spread.

Interpretation:

- D_rate = 0: The system never surprised the observer beyond the local baseline. No decisions inferred.
- D_rate > 0: Some fraction of timesteps showed surprise spikes. Decisions inferred.
- D_rate > 0.15: Frequent surprises — the system is chaotic or highly dynamic.
- D_mag: How surprising were the decision events, in MAD units. Higher magnitude = more dramatic transitions.

B.6 Derived Narratives

From M, L, and D, the observer constructs higher-order interpretations. These are not additional measures — they are patterns in the three primary measures.

Decision timing (D_centre)

```
IF decision_times is not empty:
    D_centre ← mean(decision_times) / T
ELSE:
    D_centre ← 0.5
```

D_centre near 0 = decisions concentrated early ("the system figured things out up front").
D_centre near 1 = decisions concentrated late ("the system destabilised late").
D_centre ≈ 0.5 = decisions distributed uniformly ("no temporal direction").

Character classification

```
IF D_rate > 0 AND L > 0.05 AND D_centre < 0.4:
    character ← "purposive"          # front-loaded decisions + learning
```

```
ELSE IF D_rate > 0.15 AND L < 0.01:
    character ← "chaotic"           # frequent surprises, no learning
ELSE IF M < 0.001 AND L < 0.001 AND D_rate < 0.001:
    character ← "static"            # nothing happening
ELSE IF D_rate > 0 OR L > 0:
    character ← "mixed"             # some combination
ELSE:
    character ← "inert"             # minimal activity
```

Goal-directedness (derived, not measured)

The observer infers "goal-directed behaviour" when:

- L > 0 (the system is becoming less surprising — "converging")
- D events cluster early (D_centre < 0.4 — "decisions were front-loaded")
- M > 0 (the system moved persistently — "it ended up somewhere different")

This combination produces the narrative: "The system started uncertain, made decisions early, became more predictable, and settled into a new regime." The observer calls this "working toward a goal." The system followed local dynamics.

M∩L∩D overlap

The geometric mean of normalised M, L, and D provides a single "apparent intelligence" score:

```
IF M > 0 AND L > 0 AND D_rate > 0:
    overlap ← (M × L × D_rate) ^ (1/3)
ELSE:
    overlap ← 0
```

This is not a definition of intelligence. It is a summary statistic for the region of M/L/D space where all three prediction-performance characteristics are simultaneously active — the region the observer labels "interesting."

B.7 Parameter Summary

Parameter	Symbol	Value	Used in
Prediction window	w	10 (5 for sorting)	All measures
Adaptive cap	—	max(2, T/10)	Prevents window > 10% of run

Parameter	Symbol	Value	Used in
Emission skip	—	5 timesteps	Early partition starts at t=5
Partition	—	Thirds of valid timesteps	M, L comparisons
Activity floor	—	1% of early activity	L gate (robust: 0/256 CA rules change at 0.1%–5%)
Decision threshold	—	2.0 MAD units	D detection
Decision window	—	min(15, n_valid/4)	D local baseline
Numerical floor	ε	10^{-10}	Division safety

B.8 Computational Complexity

For an emission stream of length T with d features:

- **Prediction:** O(T × w × d) — rolling mean at each step
- **M_pred:** O(T × d) — two passes over the emission stream
- **M_cohen:** O(T × d) — mean and variance per feature
- **L:** O(T × d) — comparison of early and late surprise
- **D:** O(T × w_d) — rolling median over decision window

Total: O(T × d × max(w, w_d)). For our largest substrate (256 CA rules × 300 steps × 4 features), total analysis completes in under 60 seconds on a single CPU core.