The Predictive Lock: A Cell-Based Adaptive Gating System for Personalized Trade Entry Decisions

Technical White Paper v1.0

Drive By Numbers (DBN) Trading Platform

Author: DBN Research

Date: April 2026

Abstract

This paper presents the Predictive Lock, a cell-based adaptive gating system deployed within the Drive By Numbers (DBN) trading platform that computes personalized go/no-go verdicts before trade entry. The Lock operates over a three-dimensional weight space defined by the Cartesian product of 11 ICT/SMC setup types, 5 trading sessions, and 4 market state regimes, yielding a 220-cell weight matrix initialized through a structured calibration procedure comprising a 50-question knowledge assessment and a 20-decision synthetic market replay. Each cell weight `W_{cell}` *evolves continuously through two complementary learning channels: an asymmetric nudging rule applied to every completed trade, and a dampened override learning mechanism that adjusts weights when traders deliberately bypass red (no-go) verdicts. The system incorporates stability guard rails including weekly swing limits, absolute weight clamping, and progressive live-data blending to prevent runaway adaptation. We describe the full architecture, the calibration protocol, the evaluation function with its fallback and blending logic, both learning rules with their mathematical formulations, and the validation methodology targeting a false-positive rate of 12% or less. We conclude with a discussion of known limitations and an open invitation for peer review from the quantitative trading community.*

---

1. Introduction and Motivation

The proliferation of retail trading platforms over the past decade has democratized access to financial markets, yet the overwhelming majority of retail traders continue to lose money. Industry data consistently shows that between 70% and 85% of retail forex and CFD accounts operate at a net loss. While numerous factors contribute to this attrition, a recurring theme in post-mortem analysis of failed trading accounts is the absence of a systematic, personalized gating mechanism that prevents traders from entering positions where their individual skill profile does not support a positive expected value.

Existing tools in the retail trading ecosystem fall broadly into two categories. The first category encompasses signal services and automated trading systems that externalize the decision entirely, removing the trader from the loop and thereby preventing any development of trading skill. The second category consists of educational platforms and journaling tools that provide retrospective analysis but offer no prospective, real-time guidance at the moment of trade entry — the precise moment when cognitive biases, emotional state, and pattern-recognition failures converge to produce poor decisions.

Neither category addresses the fundamental problem: that a trader's edge is not uniform across all setups, sessions, and market conditions. A trader may demonstrate consistent proficiency with Break of Structure (BOS) entries during the London session in trending markets, yet exhibit persistent losses when attempting the same setup during Asian session ranging conditions. Traditional risk management tools treat all entries identically, applying uniform position sizing rules without regard for the trader's demonstrated competence in the specific context of each trade.

The Predictive Lock was designed to fill this gap. Rather than replacing the trader's judgment or merely documenting it after the fact, the Lock provides a real-time, context-specific assessment of the trader's historical performance in the exact combination of setup, session, and market state that characterizes the proposed trade. The verdict — green, amber, or red — is grounded in personal calibration data and continuously refined through actual trading outcomes. The system is not a black box: traders can inspect the full 220-cell weight matrix, review per-cell override statistics, and observe the weekly learning

summary. The Lock is, in essence, a quantitative mirror that reflects the trader's own demonstrated edge back to them at the moment it matters most.

This paper documents the complete technical specification of the Lock system as implemented in version 1.0 of the DBN platform, with sufficient detail to enable independent replication, audit, and critique.

---

2. System Architecture

2.1 The Three-Dimensional Weight Space

The Lock models a trader's edge as a function of three independent categorical dimensions. The first dimension is the **setup type**, representing the specific ICT (Inner Circle Trader) or SMC (Smart Money Concepts) pattern that defines the trade entry. The system recognizes 11 setup types: `bos` (Break of Structure), `choch` (Change of Character), `ob` (Order Block), `fvg` (Fair Value Gap), `sweep` (Liquidity Sweep), `supply` (Supply Zone), `demand` (Demand Zone), `breaker` (Breaker Block), `mss` (Market Structure Shift), `silver_bullet` (ICT Silver Bullet), and `ote` (Optimal Trade Entry). An additional implicit category, `unknown`, is used for trades where the setup cannot be automatically classified, particularly in post-hoc evaluation of live-bridged trades from external venues.

The second dimension is the **trading session**, representing the market session during which the trade is initiated. Five session categories are defined: `asian` (approximately 00:00--09:00 UTC), `london` (07:00--16:00 UTC), `ny` (New York, 12:00--21:00 UTC), `london_close` (14:00--16:00 UTC), and `off_hours` (periods outside major session overlaps).

The third dimension is the **market state**, representing the prevailing regime as determined by the platform's ICT Analyzer engine. Four states are recognized: `trending` (directional price movement with clear higher highs/higher lows or lower highs/lower lows), `ranging` (price oscillating within a defined horizontal channel), `volatile` (elevated price variance without clear directional bias), and `thin` (low liquidity conditions with irregular price action).

The Cartesian product of these three dimensions yields `11 x 5 x 4 = 220` unique cells. Each cell is identified by a composite string key of the form `setup:session:state` (for example, `bos:london:trending`), and carries a single scalar weight `W_{cell}` in the range `[20, 100]`, where higher values indicate stronger demonstrated edge.

2.2 Weight Representation and Storage

Weights are stored as a flat JavaScript object keyed by the composite cell string. This representation was chosen over a three-dimensional array for several reasons: sparse initialization (not all cells need explicit values at any point in time), direct addressability without index arithmetic, and compatibility with JSON serialization for client-side persistence via the browser's `localStorage` API. The entire weight vector, along with calibration metadata, override history, and weekly swing tracking data, is persisted under the `dbn_core` storage key as part of the centralized `behavior` namespace within the DBNCore data engine.

2.3 Verdict Thresholds

The evaluation function maps each cell weight to one of three verdicts. A score of 65 or above produces a **green** verdict, indicating that the trader's calibration data and trade history support the proposed entry. A

score between 40 and 64 produces an **amber** verdict, with the accompanying reason text differentiating between marginal edge (score 50--64: "proceed with tight risk") and weak edge (score 40--49: "consider waiting for a better setup"). A score below 40 produces a **red** verdict, which in the platform's UI disables the Buy and Sell buttons and requires an explicit override action, including a mandatory written reflection of at least 20 characters, before the trade can be placed.

The threshold values of 65 and 40 were selected through iterative testing against synthetic trade populations and represent a deliberate asymmetry: the system is more permissive in allowing trades (amber at 40) than it is in actively endorsing them (green at 65). This design reflects the philosophical position that the Lock should function as a safety net against clearly poor decisions rather than as a prescriptive signal generator.

---

3. Calibration Procedure

3.1 Design Rationale

The calibration procedure serves a dual purpose. First, it bootstraps the 220-cell weight matrix from a cold start, providing meaningful differentiation across cells before any real trades have been placed. Second, it functions as a diagnostic assessment that surfaces the trader's relative strengths and weaknesses across four competency domains, enabling the platform's recommendation engine to suggest targeted educational content.

The calibration is administered through a dedicated page (`onboarding-calibration.html`) and consists of two sequential phases: a 50-question knowledge quiz and a 20-decision synthetic market replay.

3.2 Question Bank and Quiz Structure

The question bank contains over 200 questions distributed across four categories. The **setup recognition** category (60 questions) tests the trader's ability to identify ICT/SMC patterns — BOS, CHoCH, Order Blocks, Fair Value Gaps, Sweeps, and Market Structure Shifts — from visual candlestick chart presentations rendered as inline SVG elements via the platform's `buildMiniChart()` function. Each question presents a chart excerpt with annotated price action and asks the trader to identify the pattern, the expected directional bias, or the optimal entry zone. The **session instinct** category (50 questions) assesses the trader's understanding of session timing in UTC, strategy-session alignment (which setups perform best in which sessions), and scenario-based decision-making (for example, "Given that London session has produced a false break of the Asian range high, what is the most probable ICT setup forming?"). The **risk reflex** category (50 questions) evaluates competence in position sizing, drawdown management, correlation risk, Kelly criterion application, and recovery mathematics. The **patience** category (50 questions) probes the trader's discipline around entry timing, confirmation requirements, and emotional regulation.

From this bank, the quiz engine selects 50 questions using a fixed category split: 15 from setup recognition, 12 from session instinct, 12 from risk reflex, and 11 from patience. Question selection within each category uses a seeded pseudorandom number generator to ensure reproducibility for a given seed while providing variety across calibration attempts. Options are shuffled per question to prevent positional memorization.

Each question is presented sequentially with four options. Upon selection, the correct answer is highlighted immediately with a 1.5-second auto-advance delay, providing instant formative feedback. Scores are computed as simple percentages within each category.

3.3 Synthetic Market Replay

Following the quiz, the trader enters a synthetic market replay phase comprising four scenarios: **Trend Expansion** (strong directional market with clear institutional order flow), **Choppy Range** (horizontal consolidation with false breakouts), **News Shock** (sudden displacement event followed by retracement), and **Reversal Setup** (exhaustion pattern transitioning to opposite directional bias). Each scenario is rendered as a series of five opportunities presented through the platform's price generation engine with forced regime parameters and volatility overrides appropriate to the scenario type.

For each opportunity, the trader selects one of three actions: **Take** (enter the trade), **Skip** (explicitly decline the opportunity), or **Wait** (defer the decision pending further confirmation). Each decision is scored against an optimal action determined by the scenario's ground-truth conditions. The scoring captures the trader's ability to read market state in real time, complementing the static knowledge assessment of the quiz phase.

3.4 Weight Initialization Formula

Upon completion of both phases, the calibration engine computes per-category scores and maps them into the 220-cell weight vector using a weighted blend formula. Let `S_{setup}` denote the setup recognition score (0--100), `S_{session}` the session instinct score, `S_{state}` the average score across replay scenarios that match the given market state, and `S_{patience}` the average of the patience and risk reflex scores. The initial weight for each cell is computed as:

$$W_{cell}^{(0)} = 0.40 \times S_{setup} + 0.30 \times S_{session} + 0.15 \times S_{state} + 0.15 \times S_{patience}$$

This formula assigns the greatest influence (40%) to setup recognition, reflecting the empirical observation that pattern identification skill is the single strongest predictor of entry quality. Session awareness receives the second-highest weight (30%), acknowledging the well-documented session-dependent nature of ICT strategies. Market state reading and patience/risk discipline each receive 15%, providing meaningful but secondary influence.

The resulting weights are written to the centralized store via `DBNCore.lock.setWeights()`, which simultaneously sets the `lockCalibrated` flag and records the calibration timestamp. The `lock:calibrated` event is emitted to the cross-page event bus, enabling other platform modules to respond to the newly available calibration data.

---

4. Real-Time Evaluation

4.1 Evaluation at Trade Entry

When a trader initiates a trade on any of the platform's execution surfaces (workspace, paper trading, or prop challenge simulator), the Lock's `evaluate(setup, session, state)` function is called with context derived from the ICT Analyzer's current readings. The `setup` parameter is extracted from the trade's auto-generated ICT tags (for example, a tag of `structure:bos` maps to the `bos` setup). The

`session` parameter is derived from the active killzone detection. The `state` parameter is inferred from the analyzer's trend assessment and recent candle volatility analysis.

Input normalization strips all non-alphabetic characters (except underscores) and converts to lowercase, ensuring robust matching regardless of the formatting conventions used by different platform modules.

4.2 Exact Lookup and Fallback

The evaluation first attempts an exact cell lookup using the composite key `setup:session:state`. If the exact key exists in the weight map, its value is used directly. If the exact key is not found (which can occur when a state parameter does not match any of the four recognized categories, or when the weight map is sparse), the function falls back to computing the arithmetic mean of all weights sharing the same `setup:session` prefix across all four state values. If no matching keys exist at all, a neutral score of 50 is returned, reflecting the system's prior that an uncalibrated cell has no demonstrated edge in either direction.

4.3 Live Data Blending

A critical feature of the evaluation function is its progressive blending of calibration-derived weights with empirical trade data. After the static weight lookup, the function queries the centralized trade store for all historical trades matching the proposed setup and session. If at least 5 matching trades exist, the function computes the empirical win rate `WR_{live}` as a percentage and blends it with the calibration weight using a sample-size-dependent blending factor:

For 5 to 19 matching trades: `W_{final} = 0.60 × W_{cell} + 0.40 × WR_{live}`

For 20 or more matching trades: `W_{final} = 0.30 × W_{cell} + 0.70 × WR_{live}`

This progressive blending ensures that the calibration quiz provides meaningful early guidance when trade data is sparse, while gracefully deferring to empirical evidence as the trader accumulates a statistically significant sample in each cell. The transition from calibration-dominant (60/40) to data-dominant (30/70) reflects the epistemological principle that observed outcomes should eventually supersede predicted outcomes, but only after sufficient evidence has accumulated.

The final score is rounded to the nearest integer and clamped to `[0, 100]` before verdict determination. The evaluation return object includes the score, verdict, human-readable reason string, cell key, calibration status, and the sample size of matching trades, providing full transparency to the UI layer.

---

5. Adaptive Learning Rule

5.1 Trade Outcome Learning

Every completed trade triggers a weight update via `lock.updateFromTrade(trade)`. The function extracts the trade's auto-generated ICT tags, session, and regime, then identifies the matching setup(s), session, and state from the respective enumeration arrays using substring matching. If any of the three dimensions cannot be resolved to a recognized value, the update is silently skipped, ensuring that only well-characterized trades influence the weight vector.

For each matched cell, the update rule applies an asymmetric nudge:

$$W_{cell}^{(t+1)} = \text{clamp}(W_{cell}^{(t)} + \text{delta}, 0, 100)$$

where `delta = +2.0` for winning trades (PnL > 0) and `delta = -1.5` for losing trades (PnL <= 0).

The asymmetry is deliberate and reflects a conservative design philosophy. The positive nudge of +2 per win and negative nudge of -1.5 per loss means that a trader must maintain a win rate above 42.9% in a given cell merely to prevent the weight from declining. At a 50% win rate, the expected per-trade nudge is `0.5 × 2.0 + 0.5 × (-1.5) = +0.25`, producing a slow upward drift of approximately 1 point per 4 trades. This deliberate sluggishness prevents volatile weight oscillation in cells with small sample sizes and ensures that the Lock builds confidence slowly, requiring sustained demonstrated edge before a cell weight reaches the green threshold of 65.

Note that when a trade matches multiple setup tags (for example, a BOS that also involves an FVG retrace), all matching cells are updated, reflecting the compound nature of ICT setups where multiple confluences support a single entry.

5.2 Convergence Properties

Under stationary conditions where a trader's true win rate in a cell is `p`, the weight will converge toward an equilibrium value determined by the balance between the positive and negative nudges. Setting the expected per-trade change to zero:

$$p \times 2.0 + (1 - p) \times (-1.5) = 0$$

Solving: `p = 1.5 / 3.5 = 0.4286`, or approximately 42.9%. At win rates above this threshold, the weight drifts upward; below it, the weight drifts downward. Combined with the live-data blending described in Section 4.3 (which directly incorporates the observed win rate), the system produces a dual-signal evaluation that captures both the trajectory of weight changes and the absolute level of demonstrated performance.

The drift rate is approximately `3.5 × (p - 0.4286)` points per trade. For a trader with a 60% win rate in a given cell, the expected drift is `3.5 × 0.1714 = 0.60` points per trade, meaning roughly 25 trades are needed to shift a neutral cell (weight 50) to the green threshold (weight 65). This pace is intentionally measured, ensuring that the Lock's endorsement reflects genuine sustained performance rather than a fortunate streak.

---

6. Two-Way Override Learning

6.1 The Override Contract

The override learning system implements a bidirectional feedback loop between the trader and the Lock. When the Lock issues a red verdict, the trader is not absolutely prohibited from trading; rather, the Buy and Sell buttons are disabled until the trader explicitly invokes an override workflow. This workflow requires a written reflection note of at least 20 characters explaining the rationale for overriding the Lock's assessment. Upon confirmation, the trade is placed and tagged with `lockEvaluation.overridden = true`, along with the cell key, score, verdict, and the full text of the override note.

When the overridden trade closes, the `processOverride(trade)` function examines the outcome and adjusts the cell weight accordingly. If the trade was profitable, the Lock was overly restrictive — the cell weight is nudged upward (loosened). If the trade was unprofitable, the Lock's caution was justified — the cell weight is nudged downward (tightened). This mechanism enables the Lock to learn from its own errors: persistent false positives (red verdicts on cells where the trader actually has an edge) are self-correcting over time.

6.2 Override Learning Rate

The override learning employs a dampened learning rate of `alpha = 0.05`, substantially lower than the standard trade-outcome nudge. The delta for an override is computed as:

$$\texttt{delta}_{\texttt{override}} = \texttt{alpha} \times \texttt{magnitude} \times 100$$

where `magnitude` is derived from the trade's realized risk-reward ratio, clamped to a maximum absolute value of 3.0. Specifically, `magnitude = min(|RR|, 3.0)` for winning trades (positive sign) and `magnitude = -min(|RR|, 3.0)` for losing trades (negative sign). The multiplication by 100 scales the result into the weight space of `[0, 100]`.

For a typical winning override with `RR = 2.0`, the delta is `0.05 × 2.0 × 100 = 10.0` points. For a typical losing override with `RR = 1.0`, the delta is `0.05 × (-1.0) × 100 = -5.0` points. The R-weighting means that high-conviction overrides (large R wins) produce proportionally larger adjustments, while small or marginal outcomes produce modest shifts. The dampened learning rate ensures that approximately 20 consistent override outcomes in the same cell are required to produce a meaningful weight shift, preventing a few lucky or unlucky overrides from distorting the Lock's calibration.

6.3 Override Statistics and Transparency

The `getOverrideStats()` function provides full per-cell transparency into the override learning process. For each cell that has recorded overrides, the function computes the total count, win count, win rate, average R achieved, current weight, and a qualitative direction label (`loosened` if the override win rate exceeds 55%, `tightened` otherwise). These statistics are surfaced in the platform's Psychology module under the Deviations Ledger tab, enabling traders to inspect which cells the Lock is learning to relax and which cells the Lock's original assessment continues to be validated.

---

7. Guard Rails and Stability

7.1 Absolute Weight Clamping

All weight updates, whether from standard trade outcomes or override learning, enforce an absolute floor of 20 and ceiling of 100. The floor of 20 prevents any cell from becoming permanently blocked: even in a cell with a dismal historical record, the Lock will issue a red verdict (score 20 < 40) rather than a mathematical impossibility. This preserves the trader's agency to override, and by extension, to trigger the override learning mechanism that could eventually rehabilitate the cell. The ceiling of 100 prevents weight inflation that would make a cell impervious to subsequent losses.

7.2 Weekly Swing Limits

To prevent rapid destabilization of the weight vector — whether from a cluster of overrides in a short period or from adversarial manipulation — the override learning system enforces a weekly swing limit of 30 points per cell. The system tracks cumulative swing per cell per ISO week (Monday start) in the `lockWeeklySwings` data structure. Before applying any override delta, the function checks whether the cumulative swing for the current week has already reached the 30-point ceiling (in absolute terms). If the proposed delta would exceed this limit, it is clamped to the remaining available swing budget. If no meaningful adjustment remains (absolute clamped delta less than 0.01), the update is silently skipped.

This guard rail is particularly important during the early weeks of trading when sample sizes are small and a string of overrides in a single cell could otherwise produce dramatic weight swings. The 30-point weekly limit means that even in the most extreme case (all overrides concentrated in a single cell, all winning with maximum R), the cell can shift by at most 30 points per week — roughly equivalent to the nudge from 15 standard winning trades.

7.3 Override History Retention

The system maintains a rolling log of the most recent 500 override events, including trade ID, cell key, timestamp, outcome, realized R, PnL, applied delta, resulting weight, and the trader's override note. When the log exceeds 500 entries, the oldest records are trimmed. This bounded retention prevents unbounded memory growth while preserving a sufficient audit trail for the platform's analytical surfaces.

7.4 Learning Rate Selection

The choice of `alpha = 0.05` for override learning deserves explicit justification. At this rate, a single override with maximum R (3.0) produces a delta of 15 points — significant but not transformative. A typical override (R around 1.5) produces a delta of 7.5 points. These magnitudes ensure that override learning is perceptible in the per-cell statistics within a few weeks of active trading, but cannot dominate the calibration-derived baseline or the standard trade-outcome learning without sustained and consistent evidence.

The asymmetric nudge in standard trade learning (`+2 / -1.5`) was selected through simulation against synthetic trade populations with known win rates. The chosen values produce intuitive convergence behavior (cells with genuine edge drift upward, cells without drift downward) while maintaining weight stability under the stochastic noise inherent in small samples. Alternative nudge ratios were tested, including symmetric `+2 / -2` (too aggressive downward for typical retail win rates of 40--55%) and `+3 / -1` (too permissive, allowing losing cells to maintain green status).

---

8. False-Positive Mitigation

8.1 The False-Positive Problem

In the context of the Lock, a false positive is defined as a red verdict issued for a cell in which the trader actually possesses a positive expected value. False positives are costly: they prevent the trader from executing profitable setups and erode trust in the system. If the false-positive rate is too high, rational traders will learn to routinely override the Lock, defeating its purpose as a behavioral safeguard.

The target false-positive rate for the Lock system is 12% or less at Day 90 post-calibration. This target was derived from a cost-benefit analysis weighing the cost of blocked profitable trades against the benefit of prevented unprofitable trades, under the assumption that the average retail trader's true edge is modest (expected win rate in genuine-edge cells between 50% and 60%).

8.2 Mechanisms for False-Positive Reduction

Three mechanisms work in concert to reduce false positives over time. First, the **live data blending** described in Section 4.3 progressively replaces calibration-derived weights with empirical performance data. A cell that was unfairly penalized by the calibration quiz (perhaps the trader guessed poorly on a few questions in a specific setup category) will be corrected as real trades accumulate in that cell and the blending factor shifts toward the live win rate.

Second, the **standard trade-outcome learning** described in Section 5.1 provides a continuous upward drift in cells where the trader wins more than 42.9% of the time. Even without overrides, a cell with a genuine edge will gradually climb toward and eventually surpass the amber/green threshold.

Third, and most directly, the **override learning mechanism** described in Section 6 provides an explicit correction channel. When a trader overrides a red verdict and wins, the cell weight increases, directly reducing the probability of a false positive in future evaluations for the same cell. The dampened learning rate ensures that this correction is gradual and evidence-based rather than reactive.

8.3 Asymmetric Nudging Rationale

The asymmetric nudge ratio of `+2 / -1.5` in standard trade learning was chosen specifically to balance false-positive and false-negative rates. A symmetric ratio would produce equal sensitivity to wins and losses, but this is suboptimal in the retail trading context where base rates of profitability are low and psychological costs of blocked profitable trades (frustration, system abandonment) outweigh the financial costs of individual losing trades (which are bounded by position sizing rules). The positive bias in the nudge ratio (`+2 > |-1.5|`) ensures that the system errs slightly on the side of permissiveness, preferring to allow marginal trades rather than block them, while still providing meaningful downward pressure in cells with persistently negative outcomes.

---

9. Validation Methodology

9.1 Primary Metric: Aligned vs. Opposed Win Rates

The fundamental validation of the Lock's predictive value is the comparison between the win rate on Lock-aligned trades (green and amber verdicts) and Lock-opposed trades (red verdicts that were overridden). The `getLockAlignmentStats()` function in the DBNCore live bridge module computes this comparison across all trades sourced from external venues, where the Lock evaluation is performed post-hoc (the Lock had no influence on whether the trade was actually taken). This design is critical: because the live bridge trades are not filtered by the Lock, the post-hoc evaluation provides an unbiased estimate of the Lock's discriminative power.

A significance threshold of 30 trades is enforced before alignment statistics are reported, preventing premature conclusions from small samples. The expected result, if the Lock provides genuine signal, is

that the aligned win rate exceeds the opposed win rate by a meaningful margin (target: at least 2 percentage points at Day 30, scaling to 5 percentage points at Day 90 as the weight vector converges toward the trader's true edge profile).

9.2 Secondary Metric: False-Positive Rate

The false-positive rate is estimated by examining the outcome distribution of overridden red trades. Specifically, the false-positive rate is approximated as the proportion of red-verdict overrides that result in profitable outcomes. If the Lock's red verdicts are well-calibrated, the override win rate should be substantially below the overall win rate — ideally below 40%, given that the red threshold corresponds to cells where the trader's demonstrated edge is weakest. A red-override win rate approaching or exceeding 50% would indicate systematic false-positive generation and would trigger a review of the threshold values and learning rate parameters.

9.3 Falsification Conditions

The Lock system includes explicit falsification conditions defined at the product level. If the bootstrap cohort's win-rate lift on Lock-approved trades is less than 2% at Day 30, the calibration quiz signal is deemed too weak to justify the system's complexity, and the quiz weighting formula should be revised. If the per-cell override win rate does not converge (in cells with 5 or more overrides) to within 10 percentage points of the cell's current weight-implied win rate by Day 60, the override learning rate should be increased from 0.05 to 0.08. These conditions are documented in the platform's internal specification and will be evaluated against production data.

9.4 Edge Map Visualization

The `getEdgeMap()` function provides a structured representation of the full weight matrix as a two-dimensional grid (setup x session), with each cell value representing the average weight across all four market states. This summary view is presented to the trader in the calibration results screen and in the mobile briefing page, enabling visual identification of strengths (green clusters) and weaknesses (red clusters). The edge map serves both as a user-facing transparency mechanism and as a diagnostic tool for platform administrators reviewing aggregate calibration quality.

---

10. Limitations and Open Questions

Several limitations of the current Lock implementation warrant explicit acknowledgment. First, the calibration quiz, while comprehensive in its coverage of ICT/SMC concepts, is a static assessment that may not accurately predict live trading performance. The relationship between declarative knowledge (identifying a pattern in a quiz) and procedural skill (executing a trade in real time under uncertainty) is well-established in the expertise literature as being weaker than commonly assumed. The synthetic market replay partially addresses this gap, but operates in a simplified environment that lacks the emotional pressure and capital-at-risk dynamics of live trading.

Second, the 220-cell weight space, while manageable in terms of storage and computation, may be excessively granular relative to the data available for most retail traders. A trader who places 200 trades over several months will have, on average, fewer than 1 trade per cell — far below the minimum required for reliable cell-level inference. The live data blending mechanism (requiring 5 trades for initial blending,

20 for full blending) mitigates this sparsity at the evaluation stage, but many cells will remain at or near their calibration-derived initial values for extended periods. Future versions may explore dimensionality reduction techniques such as hierarchical Bayesian priors, where information from related cells (for example, all BOS cells regardless of session) informs estimates in data-sparse cells.

Third, the learning rules assume stationarity of the trader's skill profile. In practice, trader skill evolves over time — often non-monotonically, with periods of improvement followed by regression during strategy changes or emotional disturbances. The current system responds to these changes through the cumulative effect of trade-outcome nudges, but the response is inherently lagged. A trader who has genuinely improved in a previously weak cell must accumulate enough winning trades to overcome the historical weight deficit. Conversely, a trader undergoing a period of skill degradation may continue receiving green verdicts in cells where the weight reflects historical rather than current performance. The platform's regression detection system (described separately in the Rating system documentation) provides a complementary signal for identifying skill degradation, but is not directly integrated into the Lock's cell-level weight updates.

Fourth, the post-hoc Lock evaluation for live-bridged trades operates with degraded context. Because trades executed on external platforms (NinjaTrader, TradingView) are not mediated by the DBN ICT Analyzer, the setup and market state dimensions are evaluated as `unknown`, leaving only the session dimension as a meaningful signal. The aligned-vs-opposed win rate comparison for live trades is therefore a test of the Lock's session-level discriminative power rather than its full three-dimensional resolution. This limitation is inherent to the read-only bridge architecture and cannot be resolved without deeper integration with external platforms.

Fifth, the system does not currently account for the sequential dependence between trades. The nudge rule treats each trade as independent, ignoring the possibility that a string of losses in a cell may reflect a temporary regime shift rather than a fundamental skill deficit. Incorporating regime-duration awareness — for example, by applying reduced negative nudges during periods identified as anomalous by the ICT Analyzer's Wyckoff phase model — is an area for future investigation.

Finally, the Lock's effectiveness is contingent on the accuracy of the auto-tagging system that classifies trades into setup types at entry time. If the ICT Analyzer misidentifies a setup (for example, classifying an Order Block entry as a BOS entry), the weight update will be applied to the wrong cell, degrading the Lock's calibration over time. The platform includes an admin audit page (`admin-analyzer-audit.html`) that compares auto-tags against manual labels with a target agreement rate of 85% or higher, but this audit capability depends on traders providing manual labels, which introduces a selection bias toward more engaged users.

---

11. Invitation for Peer Review

The Predictive Lock represents, to our knowledge, the first implementation of a cell-based adaptive gating system for retail trade entry decisions that combines psychometric calibration, asymmetric reinforcement learning, and bidirectional override feedback within a single unified framework. We recognize that the system's design involves numerous parameter choices — threshold values, learning rates, nudge magnitudes, blending factors, swing limits — that were informed by simulation and domain expertise but have not yet been validated against large-scale production data.

We invite quantitative finance practitioners, behavioral economists, and trading platform engineers to review the architecture described in this paper. Specific areas where external expertise would be particularly valuable include: the optimality of the `+2 / -1.5` asymmetric nudge ratio under realistic retail win-rate distributions; the appropriateness of the 220-cell granularity relative to typical retail trade volumes; the convergence properties of the override learning mechanism under adversarial conditions (for example, a trader who systematically overrides and trades randomly); the potential for incorporating Bayesian hierarchical priors to share information across related cells; and the design of the calibration quiz, particularly the mapping from quiz scores to initial cell weights.

Correspondence regarding this paper may be directed to the DBN Research team via the platform at drivebynumbers.com. We are committed to iterating on the Lock's design in response to both empirical evidence and peer feedback, and we intend to publish follow-up analyses as production data becomes available.

This document describes the Lock system as implemented in DBN Platform v1.0 (April 2026). Implementation details are derived from the `dbn-core.js` *centralized data engine and the* `onboarding-calibration.html` *calibration interface. All code is open for inspection in the project repository.*

This document describes the Lock system as implemented in DBN Platform v1.0 (April 2026). Implementation details are derived from the dbn-core.js centralized data engine and the onboarding-calibration.html calibration interface.