Discretization of OHLC into Swing-Level Game Pieces

Status: Synthesis Proposal (S1) **Date:** December 15, 2025 **Synthesized from:** C1, C2, C3, O1, O2, G1 drafts

---

Executive Summary

This document synthesizes six competing proposals for discretizing continuous OHLC data into discrete "game pieces" suitable for recursive market generation. After analysis, I recommend a **Fibonacci Band State Machine with Structural Event Overlay**—a hybrid that combines the interpretability and data-efficiency of level-based Markov models with explicit mechanisms for the North Star's second-order rules (frustration, exhaustion, measured moves).

The core insight driving this recommendation: **interpretability is not a luxury—it is the constraint that makes this project possible given limited data.** Every draft converges on this conclusion. Neural networks are off the table. The question is which explicit, rule-based architecture best balances fidelity to the North Star, debuggability, and implementation tractability.

---

1. Problem Statement

The Challenge

The Fractal Market Simulator must generate 1-minute OHLC data indistinguishable from real markets. The Product North Star establishes that all price movement is recursive: every large move decomposes into smaller moves obeying identical structural rules. We can already *detect* these structures; we cannot yet *generate* them.

The gap is a discrete representation—"game pieces"—that: 1. Can be sequenced by a generative model 2. Preserves Fibonacci relationships, completion semantics, and scale hierarchy 3. Enables reconstruction back to OHLC 4. Is learnable from limited data (10-15 expert annotation sessions) 5. Is fully interpretable (every generated bar traceable to explicit decisions)

Success Criteria

Criterion	Measure	Threshold
Interpretability	Every state transition maps to a single English sentence	100%
Structural Fidelity	Completion (2x), invalidation (-0.1), Fib levels enforced	No violations
Self-Similarity	Same rules apply at S, M, L, XL scales	Verified by inspection
Learnability	Transition parameters estimable from available data	<100 samples per parameter
Reversibility	Game record → OHLC → Game record round-trip	Structural events preserved
Visual Realism	Domain expert cannot reliably distinguish from real data	>70% confusion rate

What We Are NOT Solving

- Bar-by-bar tick generation (wrong granularity—structure is lost)
- News event modeling (assume this layer exists; we define injection points)
- Long-term convergence (decades-scale; out of scope per North Star)
- Sub-swing microstructure (belongs in renderer, not game state)

---

2. The Discrete Representation

2.1 State: What the Generator Knows

The minimal sufficient state for predicting the next structural move is:

```
GameState:
  scales: Dict[Scale, ScaleState]  # XL, L, M, S
  global_bar: int                  # Current time position
  news_context: Optional[NewsModifier]

ScaleState:
  frame: ReferenceFrame            # The active swing defining the Fib grid
  band: FibBand                    # Current position on that grid (e.g., 0.618-
1.0)
  dwell_bars: int                  # Time in current band
```

```
  impulse: float                  # Recent momentum (distance/bars EMA)

  # Second-order rule state
  attempts: Dict[Level, int]      # Failed tests at key levels
  frustration: Optional[Level]    # Active frustration (triggers symmetric retrace)
  exhaustion: bool                # Post-completion state (mandated pullback)
  target_pressure: float          # Stacked targets density (too-many-targets rule)

ReferenceFrame:
  anchor0: Decimal                # Defended pivot (low for bull, high for bear)
  anchor1: Decimal                # Opposite pivot
  direction: Direction            # BULL or BEAR

  def ratio(self, price) -> float:
    return (price - self.anchor0) / (self.anchor1 - self.anchor0)

  def price(self, ratio) -> Decimal:
    return self.anchor0 + ratio * (self.anchor1 - self.anchor0)
```

Key Design Decisions:

1. **Oriented Reference Frame**: Bull and bear swings share one coordinate system. Ratio increases in the expected move direction. Completion is `ratio >= 2.0`; STOP is `ratio <= -0.1`. This eliminates asymmetric handling throughout the codebase.
2. **Band-Based Position**: Price position is discretized into Fibonacci bands (0–0.382, 0.382–0.5, ..., 1.618–2.0, >2.0). Transitions occur between adjacent bands only (except logged tail events).
3. **Explicit Second-Order State**: Frustration counters, exhaustion flags, and target pressure are first-class state—not emergent side effects. This is essential for debugging and rule validation.
4. **Semi-Markov Timing**: Durations are explicit (`dwell_bars`), not implicit in transition counts. This captures decision-zone chop vs. liquidity-void snaps.

2.2 Action: What the Generator Chooses

```
Action = BandTransition | StructuralEvent | Reanchor

BandTransition:
  scale: Scale
  from_band: FibBand
  to_band: FibBand                # Usually adjacent; logged exception for tail
  duration_bars: int
  impulse: float                  # Character of this move
```

```
  rationale: TransitionRationale   # void_snap, decision_chop, exhaustion_pullback, ...
  seed: int                        # For deterministic replay

StructuralEvent:
  scale: Scale
  event_type: COMPLETION | INVALIDATION | FRUSTRATION | MEASURED_MOVE
  level: FibLevel
  metadata: Dict

Reanchor:
  scale: Scale
  new_frame: ReferenceFrame        # Established after completion/invalidation
```

Three Primitive Operations: 1. **ADVANCE**: Move from one band to an adjacent band 2. **COMPLETE**: Reach 2.0 extension; swing becomes historical 3. **INVALIDATE**: Protective level violated beyond threshold; swing removed

All complex behaviors (frustration, measured moves, exhaustion pullbacks) are *composite patterns* of these primitives with attached probability modifiers—not new atomic types.

2.3 Episode Termination

An episode ends when a swing terminates. Per the North Star, termination occurs via:

Condition	Trigger	Consequence
Completion	`ratio(close) >= 2.0`	Swing becomes reference for new structure; mandatory pullback at highest scale
Invalidation	`ratio(close) <= -0.1` (S/M) or `ratio(wick) <= -0.15` (L/XL)	Swing removed; bias may flip
Frustration	`attempts[level] >= threshold`	Symmetric retrace triggered; level blocked

The generator must guarantee eventual termination for every initiated swing. This is Church's base-case requirement: recursion without termination is undefined.

2.4 Recursion Across Scales

Principle: Big moves drive small moves.

Scale hierarchy is causal, not correlational. Implementation:

1. **Top-Down Gating**: When generating an M-scale transition, the generator receives parent context: `(L_band, L_direction, L_distance_to_target, L_exhaustion)`. This context *weights* child transitions—it does not *determine* them.

2. **Nested Filling**: Generation proceeds:

   - Sample XL move (band transition + duration)
   - Within that duration window, sample sequence of L moves conditioned on XL
   - Within each L move, sample M moves conditioned on L
   - Within each M move, sample S moves conditioned on M
   - S moves render to 1-minute bars

3. **Upward Propagation**: Child events (completion, invalidation) can trigger parent state updates, but *only* through explicit structural events—never through noise aggregation.

```
XL defines the room (price range)
    L defines furniture placement (major structures within room)
         M defines movement paths (intermediate swings)
              S defines footsteps (1-minute bar generation)
```

The fractal property: The same `ScaleState` structure and transition rules apply at every scale. What differs is: - Magnitude (XL swings are larger than S swings) - Tolerance (L/XL allow soft invalidation buffer; S/M are strict) - Extremity allowance (smaller scales can be "wilder")

2.5 Canonical Game Record

The game record is the **primary artifact**. OHLC is a rendered derivative. The record must be: - **Sufficient**: Replay produces identical OHLC given seeds - **Auditable**: Every bar traceable to specific decisions - **Compact**: Store decisions, not pixels

```json
{
  "meta": {
    "instrument": "ES",
    "tick_size": 0.25,
    "start_timestamp": 1702656000,
    "master_seed": 42
  },
  "initial_state": {
    "XL": {"frame": {"anchor0": 4800, "anchor1": 5200, "direction": "bull"}, "band": "1.0-1
    "L": {...},
    "M": {...},
    "S": {...}
  },
  "news": [
```

```
    {"t": 1280, "polarity": -0.7, "intensity": "strong", "ttl_bars": 60}
  ],
  "log": [
    {
      "t": 0,
      "type": "band_transition",
      "scale": "M",
      "from": "1.0-1.382",
      "to": "1.382-1.5",
      "bars": 45,
      "impulse": 0.6,
      "rationale": "void_snap",
      "seed": 12345
    },
    {
      "t": 45,
      "type": "event",
      "scale": "M",
      "event": "FRUSTRATION",
      "level": "1.5",
      "attempts": 4
    },
    {
      "t": 45,
      "type": "band_transition",
      "scale": "M",
      "from": "1.382-1.5",
      "to": "1.0-1.382",
      "bars": 60,
      "rationale": "symmetric_retrace",
      "seed": 12346
    }
  ]
}
```

---

3. Stochastic Elements

Where Randomness Enters

Element	Distribution	Parameters	Source
Target band	Categorical	P(to_band | from_band, zone_type, parent_context, impulse)	Estimated from data
Duration	Log-normal or Gamma	, conditioned on (distance, zone_type, impulse)	Estimated from data
Impulse	EMA update	Decay rate, sensitivity	Rule-based default
News arrival	Poisson	per scale	External input
Tail override	Bernoulli × Pareto	Rare probability, heavy-tailed magnitude	Scale-dependent

Probability Modifiers

Transition probabilities are modified by context:

1. **Zone Type**: Decision zones (1.382–1.618) increase dwell; liquidity voids (1.1–1.382, 1.618–2.0) increase snap probability
2. **Parent Constraint**: If XL is in pullback, M upward transitions are downweighted
3. **Frustration Pressure**: Failed attempts at a level increase retrace probability
4. **News Bias**: Pending news tilts toward aligned moves
5. **Target Stack**: Accumulated untouched targets increase impulsive resolution or liquidation probability

The Interpretability Constraint on Stochasticity

Every probability modifier must be: - **Named**: "frustration_penalty", "void_snap_bonus" - **Inspectable**: Current value visible in state - **Tunable**: Single parameter adjustable without retraining - **Documented**: Why this modifier, what it represents

This is non-negotiable. If we cannot explain why the generator chose a transition, we cannot debug it.

---

4. Avoiding Overfitting with Limited Data

The Data Reality

We have ~10-15 expert annotation sessions. Each session yields perhaps 50-100 labeled structural decisions. This is **radically insufficient** for learning representations end-to-end.

The Strategy: Learn Parameters, Not Structure

Component	Approach
Transition rules	Fixed by North Star (which levels are adjacent, what triggers completion)
Transition probabilities	Start with rule-based priors; refine with data
Duration distributions	Log-normal with priors; fit parameters to observed inter-event times
Second-order thresholds	Fixed by North Star (frustration threshold, exhaustion behavior)

Key Insight: The North Star already provides strong priors. We don't need to *learn* that 1.382 is a pivot or that 2.0 is completion—these are axioms. We only need to *estimate* the conditional probabilities of transitions between known states.

Parameter Budget

With ~500-1000 structural decisions in our data: - ~20 transition probabilities per scale × 4 scales = ~80 core parameters - ~10 duration parameters per zone type = ~40 parameters - ~10 second-order rule parameters = ~10 parameters

Total: ~130 meaningful parameters from ~1000 observations. This is learnable with proper regularization.

Validation Strategy

1. **Leave-one-session-out cross-validation**: Train on N-1 sessions, validate on 1
2. **Structural invariant checking**: Generated sequences must never violate hard constraints
3. **Statistical comparison**: Transition frequencies, dwell distributions, completion rates match reference
4. **Visual inspection**: Domain expert review (the ultimate test)

---

5. Forward Path

Phase 1: Foundation (Week 1-2)

Goal: Lock the coordinate system, state representation, and canonical record schema.

Deliverables: 1. Implement `ReferenceFrame` with oriented ratio computation 2. Implement `ScaleState` and `GameState` dataclasses 3. Implement `FibBand` enumeration and adjacency rules 4. Define canonical JSON log schema 5. Implement deterministic replay (log → OHLC given seeds)

Validation: Unit tests for ratio computation, band assignment, and round-trip.

Phase 2: Forward Discretization (Week 3-4)

Goal: Convert real OHLC + detected swings into game records for calibration.

Deliverables: 1. Implement forward discretizer: OHLC → log 2. Extract transition counts by (from_band, to_band, zone_type, parent_context) 3. Extract duration distributions by (distance, zone_type) 4. Validate: discretized logs reproduce detected structural events

Validation: Visual overlays showing discretized bands align with annotations.

Phase 3: Single-Scale Generation (Week 5-6)

Goal: Generate syntactically valid action sequences at one scale (suggest M).

Deliverables: 1. Implement transition sampler with categorical distribution 2. Implement duration sampler with log-normal distribution 3. Implement basic OHLC renderer (Brownian bridge between band boundaries) 4. Add invariant checking (no impossible transitions, no infinite loops)

Validation: Generated M-scale sequences pass structural invariant checks; visual inspection shows plausible level interactions.

Phase 4: Multi-Scale Coordination (Week 7-8)

Goal: Full recursive generation XL → L → M → S.

Deliverables: 1. Implement parent-child context passing 2. Implement nested filling (parent duration constrains child sequence) 3. Implement cross-scale event propagation 4. Generate complete 1-minute OHLC streams

Validation: Generated data shows proper scale hierarchy; XL moves contain coherent L/M/S substructure.

Phase 5: Second-Order Rules (Week 9-10)

Goal: Implement frustration, exhaustion, measured-move, and target-stacking rules.

Deliverables: 1. Add attempt counters and frustration detection 2. Add exhaustion flag and mandatory pullback logic 3. Add measured-move triggers 4. Add target-stack pressure computation

Validation: Generated data exhibits decision-zone chop, symmetric retraces on frustration, and exhaustion pullbacks.

Phase 6: Calibration and Tuning (Week 11-12)

Goal: Refine probability parameters using empirical data.

Deliverables: 1. Estimate transition probabilities from discretized real data 2. Estimate duration parameters by zone type 3. Tune second-order thresholds 4. Build CLI for interactive parameter adjustment

Validation: Statistical comparison of generated vs. real data; domain expert review.

Falsification Experiments

Early falsification prevents wasted effort. Run these experiments before committing to full implementation:

Experiment	Question	Kill Criterion
Band Coverage	Does the Fib band discretization capture all significant price positions?	>10% of real price time spent in undefined zones
Adjacency Sufficiency	Can we model real transitions with adjacent-only moves (plus rare tails)?	>5% of real transitions require non-adjacent jumps
Duration Stationarity	Do duration distributions hold across regimes?	Kolmogorov-Smirnov test fails on held-out data
Hierarchy Coherence	Does parent conditioning produce sensible child behavior?	Visual inspection shows parent-child conflicts

Definition of "Done"

The discretization is complete when:

- ☐ Every state transition maps to one English sentence

- ☐ Canonical log round-trips through OHLC with structural fidelity
- ☐ Generator produces sequences passing all structural invariant checks
- ☐ Transition frequencies match reference data within 15%
- ☐ Duration distributions match reference data (KS test $p > 0.05$)
- ☐ Domain expert rates >70% of generated samples as "could be real"
- ☐ Documentation enables extension without rediscovering context

---

6. Risk Controls

Detecting Drift from Fidelity

Indicator	Detection	Response
Structural violation	Invariant checks in generator	Immediate crash with trace
Distribution drift	Periodic statistical comparison	Re-calibrate parameters
Visual artificiality	Expert review sessions	Identify and log specific patterns
Regime blindness	Test on held-out regimes (2022 vol, 2019 low vol)	Regime-specific parameter tables

Detecting Broken Fractal Assumptions

The fractal assumption (same rules at all scales) could be wrong. Detect via:

1. **Scale-specific validation**: Run structural checks per scale, not just aggregate
2. **Cross-scale coherence**: Verify parent-child relationships are bidirectionally consistent
3. **Extremity gradient**: Verify smaller scales show higher variance (as North Star predicts)

If fractal assumption breaks, consider scale-specific rule variants (an extension, not a redesign).

Handling Weak News Model

The current design assumes a news model providing (polarity, intensity, timing) streams. If this model is weaker than expected:

Fallback 1: Treat news as uniform random perturbations with configurable frequency/magnitude. This produces variety without semantic content.

Fallback 2: Remove news injection entirely; generate "quiet market" data that obeys structural rules without external catalysts. This is still useful for validation.

Fallback 3: Let users manually inject news events into game records for scenario analysis.

The core discretization remains valid regardless of news model quality—news modifies probabilities, it doesn't define the state space.

---

7. Failure Modes and Uncertainties

Known Unknowns

1. **Volatility Regime Dynamics**: The current design uses impulse EMA. Real volatility clustering may require more sophisticated dynamics (GARCH-like). Monitor and extend if needed.
2. **Cross-Scale Timing**: When parent commits to a duration, children must fill it. What if child dynamics naturally want longer/shorter? Current design uses soft constraints; may need adjustment.
3. **Motif Variety**: Brownian bridges may feel repetitive. May need small motif library for OHLC rendering. Defer until visual inspection demands it.
4. **Off-Grid Structure**: Real markets occasionally exhibit price levels that don't align with Fib bands. Current design treats these as noise. Monitor whether this loses important signal.

What Could Go Wrong

Failure Mode	Symptom	Mitigation
Too rigid	Generated data looks mechanical	Add noise in renderer; expand tail probability
Too random	Generated data loses structure	Tighten transition constraints; increase invariant strictness
Scale drift	Lower scales don't respect parents	Strengthen parent conditioning weights
Parameter instability	Small data changes cause large output changes	Use stronger priors; Bayesian smoothing

Honest Assessment

This approach is **not** a guarantee of success. It is the **most likely path** given our constraints. The key risks are:

1. **Visual realism may lag**: The first generated data will probably look artificial. This is expected and acceptable if structurally correct. Realism comes from iteration.
2. **Calibration is manual**: We will spend significant time adjusting parameters by hand, looking at charts, making judgment calls. This is not a bug—it's how expert systems are built.
3. **Scope is large**: The full implementation touches data loading, swing detection, state management, generation, rendering, and validation. Phased delivery with early falsification reduces risk but doesn't eliminate it.

Appendix 1: Document Synthesis

Summary of Drafts Reviewed

Draft	Primary Recommendation	Key Strength	Key Weakness
C1	Pure stochastic grammar (L-system)	Deepest interpretability analysis; explicit rule traceability	Grammar may be too regular; rendering underspecified
C2	Level-Grid Markov Model	Clean state definition; explicit tenets	Less attention to second-order rules
C3	Level-Graph State Machine + swing bookkeeping	Best separation of concerns (generator vs bookkeeper vs renderer)	Complex three-layer architecture
O1	Fibonacci Band State Machine	Compact; Andy Grove-style decisiveness	Light on recursion details
O2	HSMLM with Intent/Attempt augmentation	Best second-order rule handling; explicit volatility treatment	Most complex state
G1	Recursive Structural Grammar (FSM + Grammar hybrid)	Clear "Game Piece" definition (TargetedMove)	Less implementation detail

Common Ground (Unanimous or Near-Unanimous)

1. **Interpretability is the primary constraint.** Every draft, without exception, prioritizes interpretability over compression. Several drafts ex-

plicitly reject neural networks due to data scarcity and debugging requirements.

2. **Fibonacci bands are the coordinate system.** All drafts use Fib levels as the discrete state space. Position is always relative to an active reference swing, never absolute price.

3. **Top-down causality is causal, not correlational.** All drafts enforce that XL/L constrain M/S, never the reverse (except through explicit structural events).

4. **State machine (or equivalent) is the core abstraction.** Whether called "Markov model," "FSM," or "band machine," all drafts converge on discrete states with probabilistic transitions.

5. **The canonical record stores decisions, not OHLC.** All drafts agree that the game log should be replayable and auditable—OHLC is a rendering artifact.

6. **Limited data requires rule-based priors.** All drafts acknowledge that parameters can be tuned, but structure must be defined by the North Star, not learned from data.

Key Disagreements

Topic	Position A	Position B	Resolution
Grammar vs State Machine	C1: Pure grammar is more elegant and inherently recursive	C2, C3, O1, O2: State machine is simpler and more debuggable	State machine as core, with grammar-inspired rule organization
Explicit intent/attempts	O2: Decision zones need explicit attempt tracking	C2, C3: This is bookkeeping, not state	Include as state (O2 is correct—chop is structural, not noise)
Volatility treatment	C3, O2: Impulse/volatility is first-class state	C1, C2: Volatility is a duration modifier only	Include as state but keep simple (EMA, not GARCH)
Motif library	O2, G1: Motifs improve realism	C1, C2, C3: Motifs risk becoming a grab-bag of special cases	Defer to renderer layer; use only if visual inspection demands

Blind Spots Across Drafts

1. **News modeling**: All drafts hand-wave the news stream. "Assume it exists" is the consensus, but injection points and interaction with structural

rules need more specificity.

2. **Validation methodology**: Statistical comparisons are mentioned but not detailed. What exact metrics? What thresholds?
3. **Existing codebase integration**: Drafts define representations but don't specify how they integrate with `SwingDetector`, `ScaleCalibrator`, `BarAggregator`, etc.
4. **Error handling**: What happens when the generator encounters an impossible state (e.g., parent says go up, all up transitions blocked)?

How This Proposal Resolves Conflicts

1. **State machine as core, grammar as organization**: The recommendation uses a Fibonacci Band State Machine (C2/C3/O1 convergence) but organizes rules in interpretable named blocks with documented probabilities (C1 grammar influence).
2. **Explicit second-order state**: Following O2, the recommendation includes attempt counters, frustration flags, and exhaustion state directly in `ScaleState`. This is worth the complexity because decision-zone behavior is central to realism.
3. **Volatility as impulse EMA**: Following C3 and O2, volatility is a first-class state variable, but kept simple (EMA) rather than sophisticated (GARCH). Extend later if needed.
4. **Motifs deferred to renderer**: Following the conservative consensus, motifs are not part of the core representation. If Brownian bridges feel too uniform, add a small motif library to the OHLC renderer—but keep it separate from the game state.
5. **News as injection points**: Define explicit injection points (probability modifiers) without requiring a full news model. The design remains valid if news is weak or absent.

What Each Draft Contributes Uniquely

Draft	Unique Contribution
C1	The "interpretability imperative" argument (strongest case for rejecting neural networks)
C2	The "tenets as explicit tiebreakers" methodology (adopted here)
C3	The three-primitive action space (ADVANCE, COMPLETE, INVALIDATE)
O1	The week-by-week sequencing plan with concrete milestones

Draft	Unique Contribution
O2	The directed reference frame (oriented ratio that works for bull and bear)
G1	The “TargetedMove” framing (every move has an intended destination)

What Each Draft Misses

Draft	Gap
C1	Under-specifies OHLC rendering; grammar may be too abstract for debugging
C2	Light on second-order rules (frustration, exhaustion); may feel too “clean”
C3	Three-layer architecture may be over-engineered for initial implementation
O1	Sparse on recursion mechanics; how exactly does parent constrain child?
O2	Most complex state; may be hard to implement incrementally
G1	Least detailed; needs more implementation specificity

Document version: 1.0 Created: December 15, 2025