Theorems as Instrument Design: Formal Guarantees and Statistical Contracts in a Generative Musical Instrument

Evan Tabak Atlas
Independent · ORCID 0009-0007-7374-2338
Evanatlas@gmail.com

> Draft prepared for submission to Computer Music Journal / Organised Sound (alternate venue: NIME).

ABSTRACT

Generative instruments fail differently from other software: their worst defects are not crashes but slow statistical lies — a melodic register drifting off the usable range over minutes, variety collapsing into repetition, multi-scale structure whitening into noise while every individual note remains legal. None of this is visible in a demonstration or caught by example-based testing. We argue that the playability claims of a generative instrument should therefore be treated as *design theorems*: stated precisely as properties of the underlying stochastic process, proved formally where the mathematics is tractable, and mechanized as continuous-integration gates in every case — with statistical properties published as measurement protocols whose acceptance bands are derived once and then frozen. We develop the argument through the Fenophone, a browser-based instrument built on a Markov-switching multifractal cascade, whose engine enforces this discipline end to end: a register-gravity stability theorem with a closed-form stationary deviation and a mandatory "two-minute drift" gate; layer-invariance obtained by construction through per-layer seed substreams; monotone perceived controls verified by behavioral sweeps; and a detrended-fluctuation-analysis contract applied to every shipped and future community sound pack. We discuss what generalizes to other stochastic-process instruments, what does not (taste), and what the discipline costs.

1. The silent-failure problem

A generative instrument is a stochastic process wearing an interface. The player's experience of it — whether it feels steerable, whether it stays musical over a long session, whether the same gesture does the same thing twice — is a property of the process, not of any single output — and that is a testing problem conventional software practice does not address. A unit test checks a trajectory: given this input, that output. The failures that actually end generative instruments are properties of *distributions over trajectories*, and they are silent.

Three failure classes recur. The first is **drift**: a state variable whose first moment wanders. The canonical example, and the one this paper treats at length, is melodic register. A pitch-contour random walk with momentum sounds plausible for a minute; without mean reversion it is an undamped random walk, and — in the words of the Fenophone specification — it "floats off the usable range within minutes" (Fenophone_Product_Spec_v4.md, §3.5). The engine's own module documentation calls this "*the single most common silent way this kind of instrument fails*" (`engine/crates/core/src/contour.rs`). The second is **degeneracy**: the collapse of variety. An innocent-looking data change can pin a generator's onset pattern to a constant or an exactly periodic template while every emitted note remains individually legal. The third is **destroyed long-range structure**: the specific statistical signature the generator exists to produce (in our case, 1/f-family long-range correlation across time scales) can be whitened or trivialized without violating any per-note or per-bar constraint.

What unites these failures is their invisibility at demonstration length. Any thirty-second window of a drifting walk is unremarkable; the drift is a property of the two-minute horizon. A pack whose rhythm has collapsed to its authored bar template sounds *fine* — it is precisely as good as its template, forever. The failures are (i) not local in time, (ii) not visible in any single short run, and (iii) distributional rather than functional. Conventional testing — run the program on examples, inspect outputs — is structurally blind to all three.

There is a second-order version of the problem. When developers *do* test statistical properties, the tests are typically flaky — a threshold on a statistic over a fresh random run occasionally fails by chance — and a flaky gate trains everyone to ignore it; loosening it until it never fires converts it into decoration. Our position is that both problems have one architectural answer: make the process deterministic enough that a statistical claim over a pinned seed panel becomes a *deterministic function of the repository*, then treat each playability claim as a property with a proof obligation and a mechanized gate.

2. The method: property, theorem, gate, frozen band

The discipline we describe has four steps, applied to every playability claim the instrument makes.

1. **State the claim as a property** of the stochastic process, in the process's own terms — not "the melody sounds centered" but "the pitch-height

process has a bounded stationary range and remains within `center ± span` over any two-minute run at any legal parameter setting."

2. **Prove it as a theorem where tractable.** Linear recurrences, structural independence arguments, and monotone compositions are cheap to prove and pay for themselves: the theorem says *why* a constraint exists and where the safe parameter region lies, before any test runs.
3. **Mechanize it as a CI gate, always** — including where a theorem exists, because the theorem is about the idealized model and the gate is about the shipped implementation, and including a *negative test* proving the gate has teeth (a deliberately broken configuration must fail).
4. **For statistical properties, publish the measurement protocol and freeze the acceptance bands.** Measure the shipped corpus under a pinned protocol, derive bands with stated margins, and commit both protocol and constants; the derivation tool stays in the repository so the constants can be re-derived and audited.

The specification compresses step 4 into a phrase we adopt as a slogan: the output's statistical character is "*measured*, never assumed" (Fenophone_Product_Spec_v4.md, §14).

Three architectural moves make the method workable.

Determinism first. Everything that decides *which notes happen* — cascade switching, the harmony-graph walk, per-voice contour walks, scheduling, quantization — is computed in integer and Q16.16 fixed-point arithmetic with a single named integer PRNG (xoshiro256**) and a fixed per-tick draw order, so the event stream is bit-identical on every CPU, browser, and operating system (spec §3.6; Standing Rule 1 in PROMPTS.md). Each stochastic entity draws from its own *seed substream*, `master_seed ⊕ offset`, the offset a pure function of the entity's identity. Transcendentals run only at control-change time, in fixed point, quantized into integer thresholds; the cascade's per-tick hot path is one integer comparison, `next_u64() < threshold` (`engine/crates/core/src/cascade.rs`). (The renderer downstream is floating-point DSP held to a *published tolerance* rather than bit-identity — a deliberately weaker, separately stated guarantee; spec §3.6.) The payoff is total: every gate in this paper, statistical ones included, evaluates a deterministic function of the repository contents; gates cannot flake, so they can be strict.

Float oracles quarantined in tests. The no-floating-point rule is enforced by construction: a lint denies float arithmetic in the event-layer crate, and a test (`engine/crates/core/tests/no_float.rs`) scans its `src/` for `f32`/`f64` tokens and float literal suffixes, making any float a build failure. But analysis often

needs floats — a stationary standard deviation needs a square root; DFA needs logarithms and least-squares fits. So closed-form oracles and estimators live in test code and in the float-legal contracts crate, never in the event layer, which never computes anything a platform math library could disagree about.

Statistics as contracts over every artifact. The instrument's musical content is data: versioned, schema-validated "scaffold" packs of pitch pools, harmony graphs, rhythm grids, and voice presets (spec Appendix A; contracts/README.md, Standing Rule 6). Ten invariants (A.8) gate every pack: schema validity; voice-leading on every graph edge under every mode; non-empty legal pools with a consonance floor at maximum intensity; graph reachability; register safety over the two-minute drift run; rhythmic-coherence bounds across the parameter sweep; lane-mapping monotonicity; clip-free/no-NaN rendering; asset provenance; and spectral structure. One entry point, `check_pack` (`engine/crates/contracts/src/scaffold.rs`), runs all ten without short-circuiting — the same call for a shipped pack and for future community content. This is design by contract extended to *distributional postconditions on the emitted note stream*: a pack is accepted if and only if the process it induces has the right statistics.

The four case studies below walk the spectrum from fully provable (§3), through correctness by construction (§4) and part-theorem, part-measurement monotonicity (§5), to a frozen statistical contract (§6).

3. Case study 1: the register-gravity theorem

Each voice in the Fenophone carries a continuous pitch height `h` in semitone (MIDI) space, advanced once per scheduler tick and snapped to the nearest legal pitch at each note onset. The walk is (spec A.5; `engine/crates/core/src/contour.rs`):

```
h_{t+1} = h_t + φ·(h_t − h_{t−1})        // momentum / "Flow" memory
              − γ·(h_t − center_voice)    // register gravity (mean reversion)
              + σ·ε_t                     // step noise
```

Here `φ` is the momentum coefficient, exposed as *Flow* through the transfer `φ = 0.97·x` and clamped strictly below 1 (in fixed point, one Q16.16 unit below 1.0); `γ` is the mean-reversion strength (spec default 0.05–0.15 per step); `σ` is the step-noise scale in semitones (default 0.6–1.2); and `ε_t` is an integer approximate-Gaussian — an Irwin–Hall sum of twelve uniforms minus six, chosen because twelve gives unit variance exactly, with bounded support and a cost of exactly twelve PRNG words (`engine/crates/core/src/gaussian.rs`). A

Box–Muller transform would import `sin` and `log`, whose last bits differ across math libraries and would break bit-identity.

3.1 The theorem

Writing `d_t = h_t − center` turns the walk into a linear second-order recurrence:

$$d_{t+1} = (1 + \varphi - \gamma)\cdot d_t - \varphi\cdot d_{t-1} + \sigma\cdot\varepsilon_t$$

a damped, noise-driven oscillator with characteristic polynomial $P(z) = z^2 - (1+\varphi-\gamma)\cdot z + \varphi$. By the Jury stability test, both roots lie strictly inside the unit circle — equivalently, the walk has a bounded stationary range — if and only if

$$P(1) = \gamma > 0 \quad \wedge \quad |\varphi| < 1 \quad \wedge \quad P(-1) = 2 + 2\varphi - \gamma > 0.$$

The first condition is the theorem's punchline. With $\gamma = 0$ the polynomial has a root at $z = 1$: the process degenerates to an undamped random walk, unbounded regardless of how gentle the noise is. This is why register gravity is *mandatory* in the engine (Standing Rule 4): `ContourParams::validate` rejects $\gamma \le 0$ outright as its first check, and `is_register_stable` implements exactly the three Jury conditions (`engine/crates/core/src/contour.rs`). The third condition ($\gamma < 2 + 2\varphi$) is unreachable under the specified parameter ranges — with $\gamma \le 0.15$ and $\varphi \ge 0$, $2 + 2\varphi \ge 2$ — but the code guards it anyway, because the theorem says the region exists.

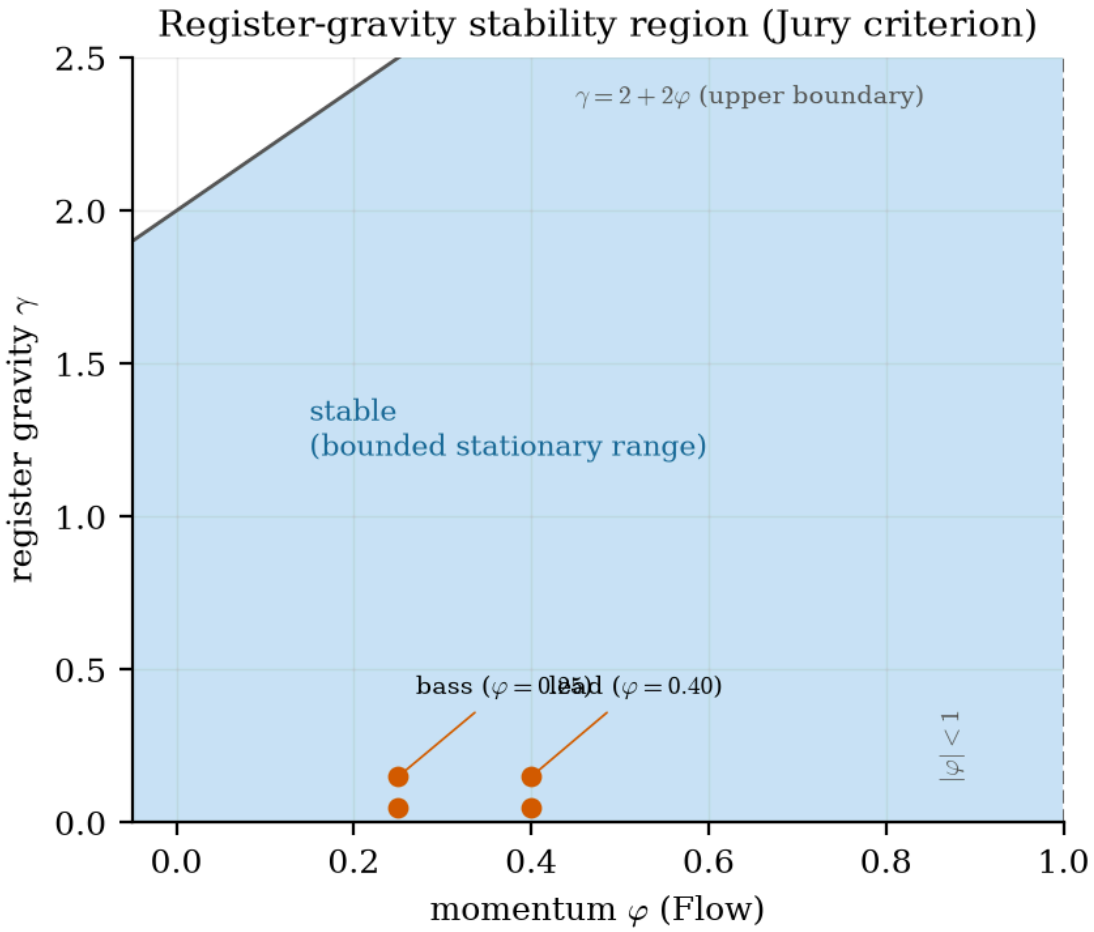

Figure 1. The (φ, γ) stability region defined by the three Jury conditions, with the default operating box (bass/lead presets, γ ∈ [0.05, 0.15]) inside it and the γ = 0 edge as the random-walk degeneracy.

Stability alone is not the playability claim: a walk can be bounded yet too loosely for its declared register band. Because the recurrence is a stationary AR(2) process driven by unit-variance noise, its stationary standard deviation has the standard closed form

$$\sigma_d = \sqrt{\sigma^2 \cdot (1+\varphi) \;/\; [\,(1-\varphi)\cdot\gamma\cdot(2+2\varphi-\gamma)\,]}$$

which the test suite verifies against simulation across the whole parameter grid (`engine/crates/core/tests/contour.rs`). A voice's register band `center ± span` is safe only when `span` is sized consistently with `(φ, γ, σ)`. The module documentation budgets the long-run maximum excursion at "roughly 5–6·σ_d"; the test suite measures ≈ 4.2–4.8·σ_d over its run length and calibrates gate bands at `span = ceil(7·σ_d)` for a non-flaky margin. The engine's default voices declare spans with margin over `6·σ_d`: the bass preset (φ = 0.25, γ = 0.15, σ = 0.60, center 40) declares span 9 against `6·σ_d ≈ 8.0`;

the lead preset (φ = 0.40, γ = 0.15, σ = 0.60, center 64) declares span 10 against `6·σ_d ≈ 8.7`.

Note the division of labor with the numerics: `σ_d` requires a square root, so it exists only as a floating-point *oracle in the tests* — the integer event layer never computes it. The theorem informs design and calibration; the shipped code contains only the three integer-comparable Jury conditions.

3.2 The gate

The "two-minute drift" test is a named, mandatory release gate from the first prototype phase onward: "register gravity keeps every voice in range over extended play. A release cannot ship if any voice exits its band" (spec §14). Mechanically (`engine/crates/core/tests/contour.rs`; `engine/crates/core/src/invariants.rs`): 4,096 ticks — two minutes at the fastest musical grid (200 BPM × 8 ticks/beat ≈ 26.7 ticks/s ≈ 3,200 ticks), with margin — at every corner of the parameter ranges (φ ∈ {0, 0.5, 0.97} × γ ∈ {0.05, 0.15} × σ ∈ {0.6, 1.2}), over 48 decorrelated master seeds, asserting the walk *never* leaves `[center − span, center + span]`. The same run, at 32 seeds per voice, is invariant #5 of the pack contract, so every scaffold — shipped or community — re-proves it for its own declared voices. The gate also asserts the band is not vacuously wide (the observed excursion must exceed `2·σ_d`), so a pinned-at-center stub cannot pass by accident.

Two negative tests give the gate teeth. A `γ = 0` voice is rejected by validation, *and* — built deliberately, bypassing validation — escapes even an absurd ±48-semitone band within the two-minute run. An undersized-span voice (φ = 0.5, γ = 0.05, σ = 1.2, span 9 against `σ_d ≈ 5.4`) passes structural validation, since `γ > 0`, yet fails the drift run — proving the gate checks the *combination* `γ/center/span`, not merely the sign of γ.

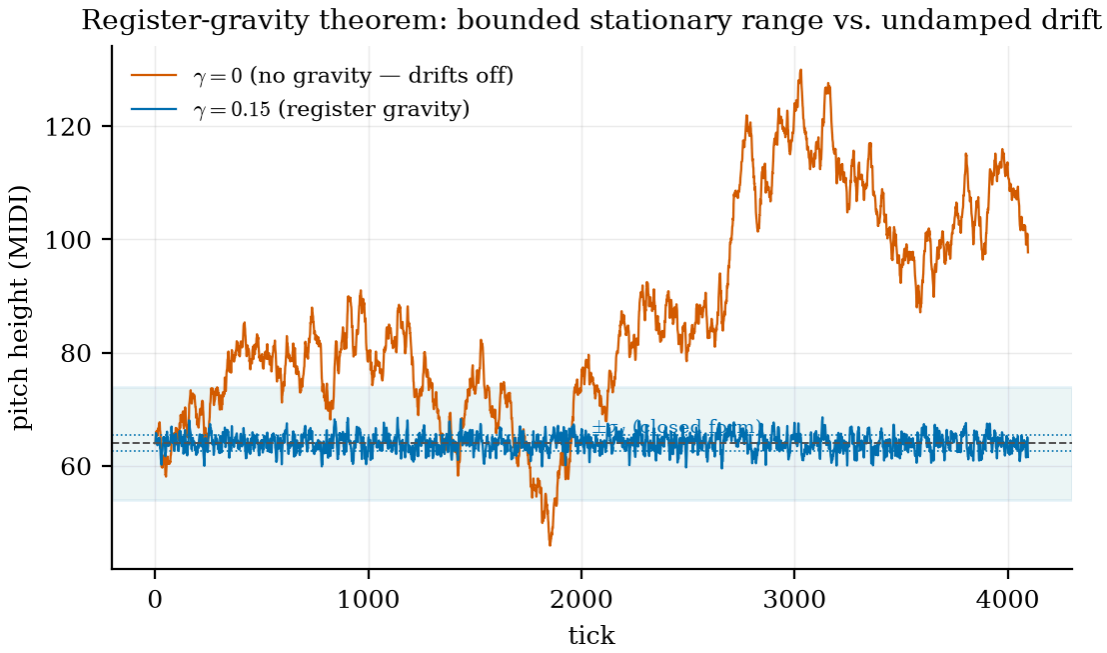

Figure 2. Two pitch-height traces from the same seed: γ = 0.15 (the walk explores but stays inside the shaded band) vs. γ = 0 (it exits and keeps going) — the canonical silent failure made visible. Illustration of the documented §3 walk equation.

3.3 What each half buys

The theorem and the gate are not redundant. The theorem answers *why*: it converts "add some pull toward the center" from folklore into a necessary and sufficient condition, locates the exact degeneracy (`γ = 0` puts a root on the unit circle), exposes a second, non-obvious boundary (`γ < 2 + 2φ`), and yields the closed form that makes band calibration principled rather than tuned by ear. The gate answers *whether, here*: it covers what the theorem idealizes away — bounded Irwin–Hall noise rather than Gaussian; saturating Q16.16 arithmetic with a fixed evaluation order and rounding rule rather than real numbers (`engine/crates/core/src/fixed.rs`, `contour_step`); finite runs; and, decisively, every concrete parameterization a scaffold author ships, including those that satisfy the theorem's hypotheses but violate the calibration. A stability theorem is here functioning as a user-experience guarantee: "the melody stays in a singable register for as long as you care to play" is, underneath, a statement about the roots of a quadratic.

4. Case study 2: layer-invariance by construction

The intensity engine is a multiplicative cascade: `k` switching multipliers (k ∈ [3, 12], default 6), each toggling between a high state `m_hi` and its mirror `m_lo = 1/m_hi` at a rate geometrically spaced across a player-controlled span, read through a fixed band map into four control lanes

(`engine/crates/core/src/cascade.rs`). The player can change `k` live. The playability property is continuity: *changing the number of layers must never perturb the layers that remain.* A naive implementation — one shared PRNG stream consumed in layer order — violates this structurally: removing layer 3 shifts every subsequent draw, so layers 4..k all change their futures, and the arrangement audibly jumps.

The Fenophone makes the property true *by construction*. Each multiplier owns an independent seed substream, `master_seed ⊕ layer_seed_offset(i)`, where the offset is a pure function of the layer index — a high-bit domain tag XORed with `i` — and *never* of `k`. All twelve slots are seeded up front; changing `k` only changes how many are active. A deactivated layer takes no draws, so its substream freezes and later resumes from the same point. The property "a layer's switch sequence depends only on `(master_seed, i)`" is not emergent behavior needing statistical verification; it is a consequence of the state layout, admitting a *direct, exact* test: two cascades built at k = 5 and k = 10 from the same master seed are asserted to hold bit-identical per-layer substreams, each also equal to an independently constructed oracle substream; a cascade shrunk from k = 8 to 6 and regrown is asserted to freeze layers 7–8 (their substreams must not advance) and to *resume* — not restart — layer 7 on regrowth (`engine/crates/core/tests/cascade.rs`).

The dividend arrives with the instrument's signature gesture. "Pinning" a layer — grabbing it and holding its state, the conducting interaction of spec §4.2 — is a per-slot override under which the layer takes *no draw* and never toggles: "its substream freezes, exactly as a deactivated layer's does, and resumes on release" (`engine/crates/core/src/cascade.rs`). Because the pin reuses the already-tested freeze/resume mechanism, the gesture inherits the invariance property for free, and its tests are again exact: an unpinned cascade is bit-identical to the pre-pin implementation; a pinned layer's substream does not advance and resumes correctly; and live pins, recorded as held overrides in the deterministic state vector, reproduce bit-for-bit under re-run (`engine/crates/core/tests/automation.rs`). The per-voice contour walks carry the same substream discipline, which is why adding or removing a voice never disturbs another voice's line — a tested §14 invariant rather than a hope.

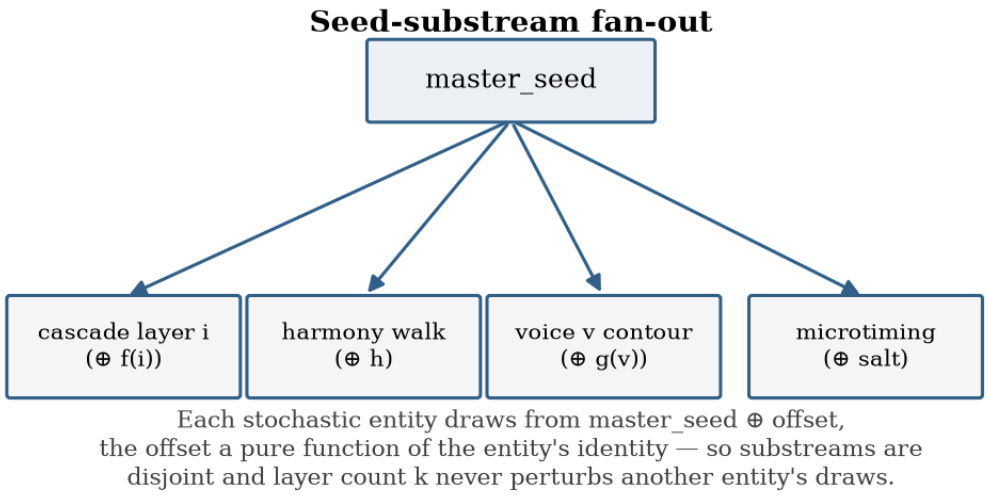

Figure 3. The seed-substream architecture: a master seed fanned out by XOR with per-entity offsets (cascade layers, harmony walk, per-voice contour, microtiming), so substreams are disjoint and the layer count never perturbs another entity’s draws.

The lesson generalizes: push the property into the *representation* — index randomness by stable identity, not consumption order — and a distributional test obligation becomes an equality assertion. Construction is the cheapest theorem.

5. Case study 3: monotone perceived controls

The specification’s controllability section opens from a confession: a multifractal process is multiplicative and naturally bursty, so “without discipline, ‘every knob is real’ could mean ‘every knob is a dice roll’” (spec §3.7). The design principle that answers it is blunt: “A move you cannot predict or reverse is a bug, gated in §14” (spec §2). The property is *monotone perceived response*: a step increase in a control produces a non-decreasing trend in the perceived quantity that control owns — more Intensity always reads as more energy — “even though the moment-to-moment micro-pattern varies. The variation is texture, not unpredictability” (spec §3.7).

The guarantee is assembled compositionally. Every control’s transfer function is monotone by specification (§4.1). Every scaffold’s lane-mapping curve is monotone — that is invariant #7 of the pack contract, checked as pure data. And the routing layer, where players rewire which band drives which voice, is a normalized non-negative mix, `drive(v) = Σ gain·band / Σ`

`gain`: a weighted average, hence monotone in each band value, so an arbitrary player rewiring cannot break the guarantee (`engine/crates/core/src/routing.rs`). Monotone pieces compose into a monotone whole — a small theorem carried by the architecture.

But composition arguments have a gap the engine names explicitly: "A scaffold whose curves are monotone can still be mis-wired so a control reads backwards; only a behavioral sweep catches that" (`engine/crates/core/src/controllability.rs`). The controllability suite is that sweep, run end to end against the running cascade and contour walks. For each of the seven steerable controls it defines a deterministic *proxy for the owned perceived feature* — Intensity → energy contrast (summed lane deviation from the ½ rest point); Spread → time-scale separation (the gap between the fastest and slowest layers' realized switch activity); Change Rate → total switch flips; Layers → the active multiplier count; Flow → melodic-run persistence (same-direction consecutive steps); Level → the dBFS gain; Tempo → the realized tick rate — sweeps it across its range with the others at a neutral baseline, and asserts the proxy is non-decreasing within tolerance *and actually responds* ("a control whose owned feature never moves is not 'real'"). Key is excluded by design: a transposition owns no monotone scalar feature; its correctness lives in the transposition-invariant harmony invariants.

Two features of the suite bear on the method. First, some proxies are backed by small proofs: because a layer's switch sequence depends only on the rates and never on `m_hi`, raising Intensity rescales lane contrast around an identical net-state stream for a fixed seed, making the energy-contrast proxy *rigorously* non-decreasing rather than statistically so. The statistical proxies (Flow, Spread, Change Rate) are averaged over decorrelated seed panels with a tolerance expressed in per-mille of the sweep's observed range plus an absolute rounding floor — small enough that a genuine reversal fails, large enough to absorb sampling wiggle. All proxies are integer summaries, honoring the float quarantine, and the suite's own comments state the design constraint plainly: "a controllability gate that flaked would be worse than none." Second, the gate has teeth: mounting a deliberately backwards Intensity mapping (`m_hi = 4·(1/50)^x`, falling as the dial rises) over a real launch scaffold must fail exactly the Intensity trend while every other control and the rhythmic-coherence bounds stay green (`engine/crates/contracts/tests/controllability_suite.rs`). The suite runs over every scaffold in the repository and asserts all three launch packs are present, so the gate cannot silently lose coverage.

The same suite carries the guarantee's second half: at every control extreme, each voice's realized onset stream must respect its declared min/max onset density and maximum clumping — never fully silent, never a wall of noise — mechanizing the specification's "bounded output" clause across the whole control space, not just at the defaults.

6. Case study 4: the spectral contract

The multiplicative cascade exists for one musical reason: structure at every time scale. Layers switching across decades of rate imprint 1/f-family long-range correlation on the note stream — the statistical signature associated with music since Voss and Clarke. The gap in the first nine invariants is stated precisely in the engine: a pack "can silently destroy that — pin the onset count constant, flatten the density path — and still satisfy every event-layer invariant, because #2–#9 check *bounds*, not *correlation*" (`engine/crates/contracts/src/spectral.rs`). Invariant #10 closes the gap by gating a *distributional* property of the emitted stream on every artifact.

The measurement protocol is published in the code and worth stating in full (`engine/crates/core/tests/spectral.rs`; `engine/crates/contracts/src/spectral.rs`):

- **Operating point.** The app's documented first-run model positions — Intensity 0.8, Spread 0.65, Change Rate 0.59, Layers 1/3 (k = 6) — through the pinned `engine-defaults.json` transfers, at the scaffold's preferred tempo, default mode, root 0.
- **Series.** From the emitted notes over 8,192 ticks (≈ 17–28 minutes of music at the launch tempi): the per-tick onset-count series and the per-tick summed-velocity series.
- **Bar-template removal.** The scaffold's metric template is an *authored, deliberately periodic* trend; the bar-phase mean is subtracted, exactly as DFA practice removes a known seasonal trend, so the estimator sees the model's fluctuations rather than the grid. This step is load-bearing: unadjusted, the template saturates the fluctuation function and the raw series measure $\alpha \approx 0.39$–0.61 across the corpus; adjusted, the cascade's scaling shows cleanly.
- **Estimator.** DFA-1 (cumulative-sum profile, non-overlapping windows, per-window least-squares detrend, RMS fluctuation, log–log least-squares fit) over published scale ladders: powers of two 16–1,024 ticks for onsets, 16–256 for velocity. The multifractal extension (MFDFA) computes the generalized Hurst exponents $h(\pm 2)$, with near-zero-fluctuation windows excluded from the negative moment, and reports the width `h(-2) - h(2)`.

- **Aggregation.** Median over a pinned panel of 8 master seeds per scaffold (`base ⊕ i·stride`, the repository's decorrelation convention).

The measured medians at this point, recorded in the test suite's documentation, are α(onset) 0.768 / 0.770 / 0.695 / 0.727 and multifractal width 0.126 / 0.108 / 0.054 / 0.102 across the four measured packs (beat-driven, cinematic, warm-ambient, prototype; the shipped library has since grown, but the frozen bands are derived from these four). From these the gate bands were derived once, with stated margins, and frozen — and the margins are now *null-anchored* rather than merely corpus-relative: a committed surrogate comparison (`foundry nulls`; 99 IAAFT + 99 shuffled surrogates per seed, built from the very series the gates measure) locates what the marginal distribution alone can produce. Median α(onset) must lie in **[0.62, 1.10]** — the floor excludes white noise (α ≈ 0.5), sits 0.075 under the weakest pack, and sits ≥ 0.07 above the shuffled null's per-seed 95th percentile (α ≈ 0.53–0.55 across the corpus); the ceiling excludes a drift-dominated stream; median α(velocity) in **[0.60, 1.05]**; median width **≥ 0.025**, keeping at least 2× margin under the weakest pack. The same comparison forced a correction the method should be proud of rather than embarrassed by: the width floor turns out to be a *degeneracy* guard, not a multifractality certificate — order-destroyed surrogates of the shipped streams measure widths of 0.04–0.09 at this length, so no fixed width threshold certifies nonlinearity. The multifractality evidence is the per-pack IAAFT excess (three packs beat their null's p95 on 8/8 seeds; warm-ambient does not and is documented as consistent with linear long memory at this length), recorded in the constant's own documentation. The velocity exponent is deliberately read over the mid-scale ladder only: at 16–1,024 it measures ≈ 0.63, which the code declines to gate as "too close to the 0.60 floor to gate honestly" — a small, telling instance of the honesty norm.

The estimator itself is pinned by *analytic anchors*, not by the music: a deterministic LCG white-noise series must measure α in [0.4, 0.6] (analytic value 0.5); its cumulative sum in [1.3, 1.7] (analytic 1.5); `h(2)` must equal plain DFA α to 10^{-12}; and the bar-template adjustment must be a statistical no-op on aperiodic input (white in, white out). A second, *interior* battery pins the estimator in the gate's own operating regime with known-Hurst fractional Gaussian noise (exact autocovariance via Hosking's recursion, deterministic Irwin–Hall innovations — the float-quarantine discipline extends to the ground-truth generator): dense fGn recovers H to within 0.01 at the panel median across the interior (H = 0.55 / 0.70 / 0.85 measure 0.548 / 0.691 / 0.855); a sparse binary observation of the H = 0.70 latent reads 0.632 — persistence survives the onset-like channel with a measured, frozen, strictly *downward* attenuation of ≈ 0.07; the template-removal step adds ≈

0.001 on top of that channel; and shuffling collapses α to ≈ 0.5. The gate's numbers are thereby bounded from both sides: they cannot be estimator inflation, and their meaning is explicitly relative to this pipeline rather than an absolute latent-H readout (`engine/crates/core/tests/spectral.rs`, the interior-anchor battery). A companion sub-gate asserts a measured *directional* fact: with other dials pinned, median α(onset) falls as Spread rises on every scaffold (0.913/0.865/0.747/0.851 at Spread 0 against 0.747/0.751/0.657/0.735 at Spread 1, with no per-seed overlap) — widening the span pushes the mid-band rates higher, so the onset stream decorrelates in fewer ticks. Even a control's *spectral* consequence is under test.

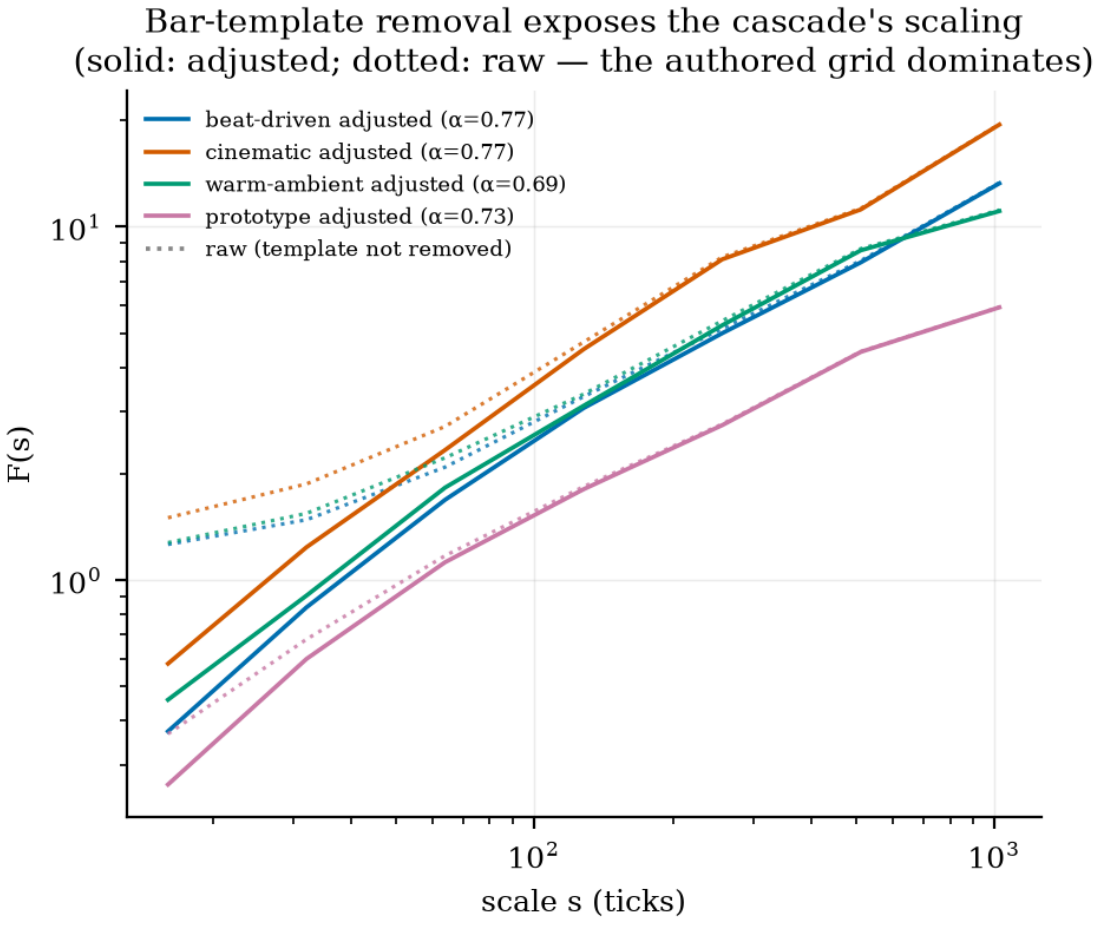

Figure 4. Log–log DFA F(s) vs. s for the launch corpus: raw onset series (dotted — the authored bar template dominates) vs. bar-template-adjusted (solid — clean cascade scaling), with the labelled α the CI-gated median. Rendered from `foundry render` (seed 0).

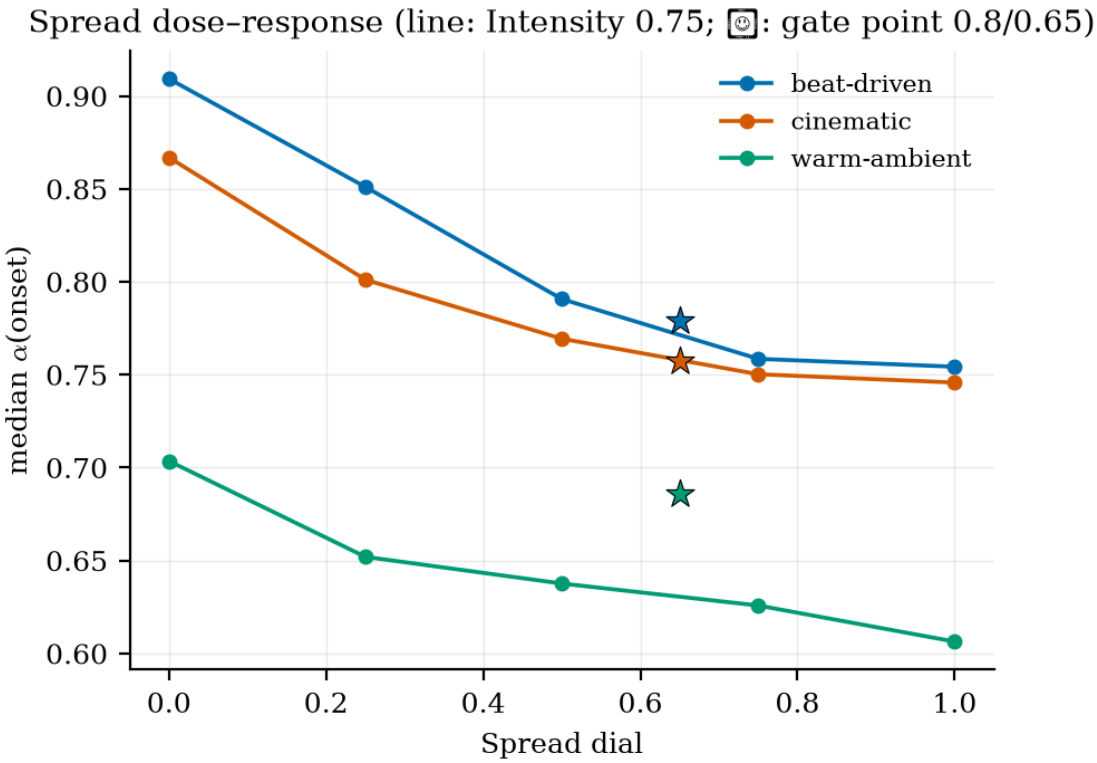

Figure 5. Median α(onset) per scaffold vs. Spread (line: Intensity 0.75; ★: the gate operating point, Intensity 0.8 / Spread 0.65), from the committed sweep medians — the gated downward ordering.

Three aspects make this gate unusual. First, it is **two-tiered**: the tight per-scaffold bands freeze the *style* of the shipped corpus, while invariant #10 proper applies a deliberately loose floor — median α(onset) ≥ 0.60 over 4,096 ticks × 4 seeds, after non-degeneracy checks (the stream must contain both onsets and silence; an exactly bar-periodic stream, whose adjusted series is identically zero, has no exponent at all and fails with a message naming the destroyed structure) — to *every* pack `check_pack` ever sees, including future community content. The invariant guards against destroying multi-scale structure; it does not police style drift. Second, it **cannot flake**: the runs under the gate are integer-only and panel-seeded, so every series is bit-identical everywhere, and the f64 estimation on top varies by a few ulps across platforms — orders of magnitude below the margins; in the code's words, "the verdict cannot flake." A distributional CI gate is ordinarily a contradiction in terms; determinism dissolves it. Third, the **measurement tool is part of the artifact**: an `#[ignore]`-marked reporter test prints the full per-seed spectral landscape so the published constants can be re-derived and audited — the reproducible-research habit of shipping the analysis script with the paper, imported into a test suite.

One duplication deserves honest mention. The DFA core exists twice — once in the engine's test suite, once in the contracts crate — because the two crates cannot share test-only code across their dependency direction. Both copies carry a "keep in sync" notice and are pinned by the identical analytic anchors, which is the mechanism that keeps duplication honest; we return to this as a cost in §7.3.

7. Discussion

7.1 What generalizes

Nothing in the method is specific to multifractal cascades. Any instrument built on a stochastic process — Markov chains over pitch, stochastic grammars, agent swarms, learned latent processes — supports the same treatment: identify the stationary or structural property behind each playability claim; prove what is linear, monotone, or structural; gate the rest under a pinned, published protocol with frozen bands and negative tests. The recipe in one line: *state the stationary property, derive the bounds, gate the tails.* The register-gravity treatment transfers almost verbatim to any melodic random walk — most have an AR(p) skeleton whose stability region and stationary variance are classical. Two properties added to the engine after this paper's first draft exercise the recipe in place: cascade-modulated pitch volatility (`pitch_volatility`) extends the register-gravity theorem itself — the same two-minute drift gate re-proves stability at the coupled step-noise ceiling σ·2^x — and multifractal microtiming adds a new frozen statistical contract of exactly the §6 kind, a dedicated timing sub-cascade whose emitted sub-tick deviations are gated for long-range correlation (DFA $\alpha \approx 0.90$ against a shuffled null ≈ 0.49). Both ship disabled-by-default on the launch packs, so neither disturbs a single number above; each is the method applied once more.

The precondition, though, is architectural and must be chosen early: determinism, in the strong form of a bit-identical decision layer with per-entity seed substreams. It cannot be retrofitted cheaply; it converts statistical claims into non-flaky gates, makes bug reports reproducible, and — not incidentally — makes the instrument learnable, "not a slot machine" (spec §1). The seed-substream discipline is the single highest-leverage decision in the codebase: simultaneously the reproducibility mechanism, the k-invariance proof, and the reason the pin gesture was correct on arrival.

7.2 What does not generalize: taste

The invariants bound failure; they do not produce art. The specification's own principle — "Taste is guaranteed by the scaffold, not hoped for" (spec

§2) — should be read carefully: the guarantee is that *authored* taste survives the model's extremes (pools stay consonant, rhythm stays coherent, structure stays multi-scale), not that a machine has taste. The scaffold data — the chord graphs, the pools, the grids — is authorship, and the theorems are silent about whether it is any good. Accordingly, the release process retains a calibrated human gate alongside the automated ones: a blind listening panel of at least eight recruited listeners scoring at least thirty random patches across all scaffolds and extremes on a fixed, versioned rubric, with a median of at least 3.5/5 required on both "would I keep this?" and "could I steer it?" (spec §14). The coexistence of the two gate families is the honest position: the mechanized gates make the human gate *tractable* (the panel never wastes time on drifting registers or dead knobs), and the human gate covers what no invariant can. Formal methods bound the instrument from below; they do not aim it.

7.3 Costs

The discipline is not free, and a methods paper should price it.

Fixed-point asceticism. Banning floats from the decision layer means owning a Q16.16 type with documented saturation and rounding, evaluating transcendentals only at control-change time, building an integer Gaussian, and exiling even a square root to test-side oracles. Every algorithm is re-derived in this vocabulary (the cascade's per-tick work collapses to one integer comparison against a precomputed threshold — elegant, but it had to be designed). The compensation is that the entire guarantee stack of this paper rests on it.

Protocol freezing. Published constants are hostages. Any change to the estimator, the operating point, or the scale ladders invalidates the frozen bands, and the tight bands deliberately freeze the current corpus's style: a legitimate future aesthetic shift will trip them and force a deliberate re-derivation and re-freeze. The repository mitigates this with the two-tier design (loose invariant vs. tight style bands) and by keeping the measurement reporter in-tree, but the tension between gating and evolving is real and permanent.

Duplication across boundaries. The verbatim DFA duplication across the crate dependency boundary is a genuine maintenance smell, accepted as cheaper than restructuring the dependency graph, and held in check by identical analytic anchors on both copies. Anchors-as-contract is a useful pattern — two implementations pinned to the same analytic ground truth can drift in text but not in behavior — but it is a mitigation, not a cure.

Judgment in tolerances. The statistical trend gates require chosen tolerances (per-mille-of-range dips, seed-panel sizes, safety multiples like the drift gate's 7·σ_d). They are principled and documented, but they are choices; a gate is only as honest as its margins, which is why writing the margin *rationale* into the constant's documentation ("0.075 under the weakest pack"; "too close to the floor to gate honestly") matters as much as the values.

7.4 Relation to existing practice

The method sits at the intersection of three traditions. From **property-based testing** (Claessen and Hughes 2000) it takes the primacy of properties over examples and the insistence on negative tests, but inverts the randomness: where QuickCheck randomizes inputs per run, these gates pin seed panels so that a statistical property becomes a deterministic regression — trading generative breadth for the non-flakiness that statistical assertions need. From **design by contract** (Meyer 1992) it takes machine-checked obligations at interfaces, extended from preconditions on data (the schema, the graph properties) to distributional postconditions on emitted streams — invariant #10 is, to our knowledge, an unusual object: a detrended-fluctuation exponent as an acceptance clause on community content. From **reproducible research** it takes the published protocol, the pinned environment, and the in-tree derivation script for every published constant. The spectral gate also inverts the oldest quantitative claim in this literature: where Voss and Clarke (1978) *observed* 1/f structure in music and proposed it as a compositional prior, here the 1/f-family character is an enforced invariant of an instrument — measured on every build, never assumed. Finally, we note a sociotechnical motivation recorded in the repository itself: the project is a solo build heavily assisted by AI coding tools, and integer-deterministic, exhaustively-gated designs were chosen partly *because* they make machine-generated test coverage trustworthy (PROMPTS.md, Standing Rule 8). We suspect this coupling — playability contracts as the safety rail for AI-assisted instrument building — will become common, and deserves study in its own right.

8. Conclusion

The claims that make a generative instrument playable — it stays in register, a gesture never perturbs what it did not touch, every knob reads monotonically, the output keeps structure at every scale — are claims about a stochastic process, and the ways they fail are silent, slow, and statistical. We have argued, through four case studies from one working codebase, that such claims can and should be engineered like theorems: stated as

properties, proved where the mathematics cooperates (a Jury condition as a user-experience guarantee), obtained by construction where the representation allows it (seed substreams), and mechanized as deterministic CI gates in every case, with statistical properties held to published, frozen measurement protocols. The precondition is determinism in the decision layer; the residue that no theorem reaches is taste, which remains authored and human-judged. What the discipline buys is a specific kind of honesty: an instrument whose playability is not a demo that went well, but a set of properties that fail loudly, in CI, before any listener is disappointed — measured, never assumed.

References

- Blackman, D., and S. Vigna. 2021. “Scrambled Linear Pseudorandom Number Generators.” *ACM Transactions on Mathematical Software* 47(4).
- Calvet, L. E., and A. J. Fisher. 2004. “How to Forecast Long-Run Volatility: Regime Switching and the Estimation of Multifractal Processes.” *Journal of Financial Econometrics* 2(1):49–83.
- Calvet, L. E., and A. J. Fisher. 2008. *Multifractal Volatility: Theory, Forecasting, and Pricing*. Academic Press.
- Claessen, K., and J. Hughes. 2000. “QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs.” In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pp. 268–279.
- Hu, K., P. Ch. Ivanov, Z. Chen, P. Carpena, and H. E. Stanley. 2001. “Effect of Trends on Detrended Fluctuation Analysis.” *Physical Review E* 64:011114.
- Jury, E. I. 1964. *Theory and Application of the z-Transform Method*. Wiley.
- Kantelhardt, J. W., S. A. Zschiegner, E. Koscielny-Bunde, S. Havlin, A. Bunde, and H. E. Stanley. 2002. “Multifractal Detrended Fluctuation Analysis of Nonstationary Time Series.” *Physica A* 316(1–4):87–114.
- Meyer, B. 1992. “Applying ‘Design by Contract’.” *IEEE Computer* 25(10):40–51.
- Peng, C.-K., S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. 1994. “Mosaic Organization of DNA Nucleotides.” *Physical Review E* 49(2):1685–1689.
- Voss, R. F., and J. Clarke. 1978. “‘1/f Noise’ in Music: Music from 1/f Noise.” *Journal of the Acoustical Society of America* 63(1):258–263.

Companion records of the Feno program (DOIs published 2026-08-31)

*Every record below is deposited on Zenodo — staged as a private draft with a **reserved** DOI 2026-08-30, **published 2026-08-31**: every DOI below is live and resolves. The deposit map is `research/zenodo/records.json`, and the source repositories are private, so each record carries its own reproduction materials. Cross-citations follow that map’s `related_identifiers`.*

- Atlas, E. T. 2026. “The Fenophone: A Musical Instrument Built on the Markov-Switching Multifractal, with CI-Verified Spectral Invariants” (Fenophone research series, paper 01). Zenodo DOI 10.5281/zenodo.22180423 (published 2026-08-31).
- Atlas, E. T. 2026. “The Markov-Switching Multifractal as a Generative Model of Musical Intensity: An Instrument, Measurements, and a Research Programme” (Fenophone

research series, paper 03). Zenodo DOI 10.5281/zenodo.22180429 (published 2026-08-31).