Executive Brief —
Shunyaya Symbolic Mathematical AI (SSM-AI)

A conservative, drop-in alignment lane that shows stability and reduces surprises — **without changing your numbers**.

Status: Public Research Release (v1.8)
Date: 27 October 2025
Caution: Research/observation only. Not for critical decision-making.

Three business questions your leadership should be able to answer

1. **Can you see the cliff before revenue falls?** When KPIs look great, does a stability lane flag growing tail risk early?
2. **Are weekly wins actually durable?** When totals tie, can you see drift that predicts next month's miss or overage?
3. **Are you paying a stability tax?** When a unit looks efficient, does a band view expose retries, latency tails, and incident risk you are not pricing?

Three AI platform questions your product teams should be able to answer

1. **Are model choices sturdy under real traffic?** When two vendors score similarly on accuracy, does a bounded chooser `RSI in (-1,+1)` reveal which one stays stable through spikes, tools, and RAG drift?
2. **Do you catch failure faster than you ship it?** Can you band and gate alignment (`RSI_env := g_t * RSI`) to quarantine regressions before they hit users or costs?
3. **Can you replay decisions exactly?** If a ranking or route changes, can you reconstruct it from stamped `U/W` roll-ups — order-invariant and vendor-neutral — without touching your original metrics (`phi((m,a)) = m`)?

What it is (one line)

A conservative add-on that leaves every KPI exactly as is and adds a small, bounded stability lane beside it: `x := (m, a)` with `a in (-1,+1)` and collapse parity `phi((m,a)) = m`; decisions read a simple chooser `RSI in (-1,+1)` and bands `(A++/A+/A0/A-/A--)` to show sturdiness — no rewrites, no PII, and replayable stamps if you want them.

Why leaders care today (outcomes over math)

- **Fewer surprises, earlier.** Stability bands surface tail risks and drift before classical KPIs move.
- **Cleaner vendor choices.** A single bounded chooser `RSI in (-1,+1)` makes cross-model comparisons fair; no calibration wars.
- **Lower run costs without guesswork.** Token, latency, and retry improvements show up as stable `A+/A++` bands; savings remain outside the lane math.
- **Quieter operations.** Band hysteresis trims alert flicker; teams act on trends, not noise.
- **Evidence you can replay.** Optional stamps and `knobs_hash` make decisions tamper-evident and reproducible, end to end.

What SSM-AI delivers (at a glance)

- **Confidence in every output.** Whether it is a model response, a forecast, or a KPI, each carries a visible stability lane that signals reliability in real time.
- **Vendor-neutral truth layer.** Adds bounded symbolic entropy (`RSI`, `a`) over existing outputs, so teams can compare models and tools without bias or retraining.
- **Zero friction deployment.** Drop-in SDK; no model retrain, no data migration, and no infrastructure rewrite.
- **Faster AI economics.** Drift detection and early failure cues help cut wasted tokens, retries, and review loops — all while leaving the original metrics intact (`phi((m,a)) = m`).
- **Ethical and auditable by design.** Optional SSM-Clock stamps and `knobs_hash` provide full replayability, offering audit confidence across the AI value chain.

Under the hood (roots, one-liners)

- **Core math (Shunyaya Symbolic Mathematics — SSM).** Two-lane numeral `x := (m, a)` with `a in (-1,+1)`. Safe map: **clamp** → `u := atanh(a)` → **sum in u** → `a := tanh(u)`. Collapse parity holds: `phi((m,a)) = m`.
- **Symbols (SSMS).** Small verb set for clean connectors: `CLAMP`, `MAP(atanh/tanh)`, `FUSE(U/W)`, `MUL_DIV(M2)`, `BAND`, `GATE`, `STAMP`.
- **Time & provenance (SSM-Clock + SSM-Clock Stamp).** Optional, tamper-evident stamps and order-invariant roll-ups in u-space. One-line stamp: `SSMCLOCK1|iso_utc|rasi_idx|theta_deg|sha256(file)|chain`.
- **Hardware (SSMH).** Same lane math on tiny fixed-point MACs: atanh-in, tanh-out, streaming `(U,W)` accumulators for deterministic, low-cost speedups.
- **Search (SSM-Search).** Lane-native retrieval: rank by bounded `RSI` while classical retrieval scores remain untouched.

What is included in SSM-AI (integration now).

Included today: SSM core lane math, SSMS verbs, SSM-Search scorer/ranker, SSM-Audit KPI overlays, optional SSM-Clock Stamp fields, and the SSM-Hardware parity spec (software↔fixed-point equivalence).

Roadmap for SSM-AI integration.
Domain-specific adapters delivered as manifest presets and evidence packs (observation-only), built on SSM modules — for example, an SSM-Chemistry adapter.

One-liner. One bounded lane, many surfaces: `clamp → atanh → sum → tanh` powers provenance, acceleration, search, and audit overlays — while `phi((m,a)) = m` keeps the original numbers pristine.

Limits & failure modes (candor box)

- **Low-signal regimes.** If the declared lens is weak or noisy, `a` will hover near `0`; treat bands as "insufficient evidence," not a verdict.
- **Poor lenses.** Badly designed lenses mislead any chooser. Publish lens recipes and stick to bounded transforms.
- **Throughput before acceleration.** Extremely high-throughput paths should use SSMH or vectorized u-fuse to avoid CPU hotspots.
- **Division policy.** Defaults are strict; if you need meadow-style semantics, declare it explicitly and test with acceptance checks.

In one line

SSM-AI makes stability visible and measurable — so your AI runs calmer, cheaper, and more trustworthy, with zero code forks.

Engineer's 1-Pager — Evaluation Guide

Kernel (never touches m).

- **Object:** `x := (m, a)` with `a in (-1,+1)`; **collapse:** `phi((m,a)) = m`.
- **Clamp:** `a_c := clamp(a, -1+eps_a, +1-eps_a)`.
- **Map:** `u := atanh(a_c)`; inverse `a := tanh(u)`.
- **Fuse (order-invariant):** `U += w*atanh(a)` ; `W += w` ; `a_out := tanh( U / max(W, eps_w) )`.
- **Weights:** `w := |m|^gamma` (default `gamma = 1`).

- **Mul/Div (M2 by default):**
    - Multiply: `(m1,a1)*(m2,a2) := ( m1*m2 , tanh(atanh(a1_c)+atanh(a2_c)) )`
    - Divide (strict, `m2 != 0`): `(m1,a1)/(m2,a2) := ( m1/m2 , tanh(atanh(a1_c)-atanh(a2_c)) )`

Chooser & gating.

- **Chooser:** `RSI := tanh( (V_out - U_in) / max(W_in, eps_w) )` (declare lens → produce `U_in, V_out, W_in`).
- **Calm gate (optional):** `RSI_env := g_t * RSI`, with `g_t in [0,1]` built from bounded drift lanes; keep gate observation-only.

Defaults (normative, copy into manifest).

- `eps_a = 1e-6`, `eps_w = 1e-12`, `gamma = 1`.
- **Bands:** `A++/A+/A0/A-/A--` (publish your numeric thresholds once).
- **Division policy:** `strict` by default; alternatives must be declared and tested.
- **Rounding:** half-even at export, `>= 6` decimals; full precision internal.

12-line pseudocode (reference).

```
a_c = clamp(a, -1+eps_a, +1-eps_a)
u   = atanh(a_c)
for each stream item:
    w  = abs(m)**gamma
    U += w * atanh(clamp(a_i, -1+eps_a, +1-eps_a))
    W += w
a_out = tanh( U / max(W, eps_w) )
RSI   = tanh( (V_out - U_in) / max(W_in, eps_w) )
RSI_env = g_t * RSI   # optional
band  = band_of(a_out)  # A++..A--
assert phi((m,a_out)) == m
```

Manifest stub (minimal).

```
{
  "eps_a": 1e-6, "eps_w": 1e-12, "gamma": 1,
  "bands": { "A++": 0.80, "A+": 0.40, "A0": -0.10, "A-": -0.40, "A--": -
0.80 },
  "division_policy": "strict",
  "rounding": { "mode": "bankers", "decimals": 6, "when": "export" },
  "gate": { "enabled": false }  # set true only if you publish Z/A recipes
```

```
}
```

Acceptance checks (drop-in).

- **Collapse parity:** `phi((m,a)) == m` after every op.
- **Order-invariance:** shuffle test on any stream yields `|a_out_diff| <= 1e-12`.
- **Division policy:** strict test near `m2 -> 0` fails fast; meadow only if declared.
- **Bounds:** always enforce `|a| < 1`, `|RSI| < 1`.
- **Determinism:** same inputs → identical `U, W, a_out, band` after rounding.

Golden vectors to wire into CI (starter set).

1. Clamp edges: `a = ±1` → clamped, no NaNs.
2. Long-path fuse: 100-step stream equals batched fuse.
3. Mul/Div (M2): `a'* = tanh(atanh(a1)+/-atanh(a2))`.
4. Bands + hysteresis: promote/demote only past declared deltas.
5. Strict division near zero: enforce policy choice.
6. Decode rerank micro (text): bounded `RSI` selects the calmer option with identical `m`.

What changes in your code.
Nothing on `m`. You attach `(a, band)` beside your existing outputs; your selection logic can read `RSI` or `RSI_env` as a bounded, comparable index.

OMP