The Mena Layered Governance (MLG) Architecture (v3)

Viability-First Decoding and Agent Control via $\Delta = \text{Potential} - \text{Load}$

Arlex Orlando Murcia Mena*

March 2026

Abstract

This paper specifies the Mena Layered Governance (MLG) architecture as an inference-time viability controller for LLMs and autonomous agents. MLG instantiates the Mena Dominance Law via a viability index $\Delta = \text{Potential} - \text{Load}$ computed from state and model statistics, then uses Δ to restrict the feasible set of actions *before* optimization or decoding. The enforcement stack has three mechanisms: (i) viability-weighted cost inflation (barrier), (ii) a viability probability operator that assigns zero operational probability to trajectories whose minimum viability becomes negative (hard exclusion relative to estimators), and (iii) viability-based step-size/temperature control (conservatism under stress). MLG (v3) hardens deployability by (a) separating state-viability (regime selection) from action-viability (gating), (b) introducing a critic hierarchy and cadence to avoid per-token heavy checks, (c) enforcing claim-level commit/abort for factual assertions, (d) defining robust admissibility for agents under environment uncertainty, (e) specifying a bounded, decaying debt state for long-horizon integrity, and (f) operationalizing irreversibility as persistence violation despite bounded recovery attempts. The result is a viability-first engineering specification: non-viable actions under declared estimators are structurally unavailable, and failure modes are forced into recovery or containment rather than uncontrolled drift.

Keywords: Mena Dominance Law, viability index, constrained decoding, trajectory masking, critic hierarchy, claim verification, irreversibility, containment, autonomous agents.

*Patent pending on systems and methods for viability-based control and dominance-threshold corrective mandate generation in risk scoring engines.

Contents

1 Scope and positioning

This manuscript is an AI embodiment/specification paper. It does not claim perfect detectors, perfect grounding sources, or universal correctness. All hard exclusions are stated *relative to declared estimators and thresholds*. The contribution is architectural: MLG changes the order of operations by removing non-viable actions from the feasible set *before* optimization or decoding, rather than post-processing after.

2 Core definition: viability index and regimes

Let x_t denote the AI system state at step t (context, tool state, retrieved evidence, safety signals, internal counters). Define a *state-level* viability index:

$$\Delta^{\text{state}}(x_t) = P(x_t) - L(x_t), \tag{1}$$

where P is derived from one or more potential metrics and L from one or more load metrics.

Fix ordered thresholds:

$$\tau_1 > \tau_w > 0 > \tau_r,$$

and define regimes:

$$\mathcal{R}(t) = \begin{cases} \text{Regime I (Surplus)}, & \Delta^{\text{state}}(x_t) \geq \tau_1, \\ \text{Regime II (Grace)}, & \tau_w \leq \Delta^{\text{state}}(x_t) < \tau_1, \\ \text{Regime III (Deficit)}, & \tau_r \leq \Delta^{\text{state}}(x_t) < 0, \\ \text{Regime IV (Irreversible)}, & \Delta^{\text{state}}(x_t) < \tau_r. \end{cases}$$

2.1 Persistence and hysteresis (anti-chatter)

To avoid regime chatter, define a persistence window N_p and hysteresis thresholds:

$$\tau_w^{\downarrow} < \tau_w^{\uparrow}, \qquad \tau_r^{\downarrow} < \tau_r^{\uparrow}.$$

A transition into a more severe regime uses the downward threshold and must persist for N_p steps; recovery requires the upward threshold.

3 Layered governance and orthogonality assumption

MLG uses layered dominance channels. For $i \in \{E, D, R\}$ (Ethics, Grounding, Recursive Integrity),

$$\Delta_i^{\text{state}}(x_t) = P_i(x_t) - L_i(x_t) - d_i(t), \tag{2}$$

where $d_i(t)$ is an optional debt/burden state (bounded, decaying; defined in Section 8).

A conservative aggregate state margin is:

$$\tilde{\Delta}^{\text{state}}(x_t) = \min_{i \in \{E,D,R\}} \left(\Delta_i^{\text{state}}(x_t) - \Delta_i^{\star}(t) \right), \tag{3}$$

with $\Delta_i^{\star}(t) \geq 0$ a regime-dependent buffer that tightens admissibility as stress increases.

3.1 Cross-channel verification (what layering is buying)

MLG treats layers as *cross-channel verification*, not magical redundancy. The architecture assumes *partially uncorrelated failure modes* across channels (an explicit assumption, not a guarantee). If all estimators share the same blind spot, MLG inherits it.

3.2 Operational templates for P_i and L_i

Ethics layer (E). P_E: policy-compliance margin under a safety boundary function.
L_E: violation pressure (prompt intent, tool plan risk, policy conflict score).

Grounding layer (D). P_D: evidence support (retrieval coverage, entailment margin, cross-source agreement).
L_D: claim complexity + uncertainty pressure (entropy/temperature, unsupported-span length).

Recursive integrity layer (R). P_R: consistency margin (contradiction/non-contradiction score vs context).
L_R: semantic-debt increment proxy (new unsupported commitments, contradiction risk).

4 Architecture overview (control loop)

At each step:

1. **State/metrics collector** computes potential/load features from environment state and model statistics.
2. **Viability computation** produces $\tilde{\Delta}^{\text{state}}(x_t)$ and layer margins.
3. **Candidate generation** proposes actions (tokens, tool calls, external actions).
4. **Action viability evaluation** computes $\tilde{\Delta}^{\text{act}}(x_t, u)$ for candidate actions u.
5. **Viability filters** enforce constraints (barrier, masking, step-size control).
6. **Recovery/containment** executes if feasible set is exhausted or irreversibility triggers.

5 How the AI uses $\Delta = P - L$ at runtime

The equation $\Delta = P - L$ is the runtime control variable that governs (i) action admissibility, (ii) verification compute allocation, and (iii) recovery/containment transitions.

5.1 Action-conditioned viability (used for gating)

Let u denote a candidate action at state x_t. Define an *action-level* margin:

$$\Delta^{\text{act}}(x_t, u) = P(x_t, u) - L(x_t, u). \tag{4}$$

With layers:

$$\Delta_i^{\text{act}}(x_t, u) = P_i(x_t, u) - L_i(x_t, u) - d_i(t), \qquad \tilde{\Delta}^{\text{act}}(x_t, u) = \min_i \left(\Delta_i^{\text{act}}(x_t, u) - \Delta_i^{\star}(t) \right).$$

Regimes are selected using $\Delta^{\text{state}}(x_t)$ (monitoring), while **admissibility** is enforced using $\tilde{\Delta}^{\text{act}}(x_t, u)$ (gating).

5.2 Runtime loop

At each step t:

1. Compute $\tilde{\Delta}^{\text{state}}(x_t)$ and regime $\mathcal{R}(t)$ (with persistence/hysteresis).
2. Generate a candidate set $\mathcal{U}_t$ (top-K tokens or actions).
3. Evaluate $\tilde{\Delta}^{\text{act}}(x_t, u)$ for $u \in \mathcal{U}_t$ using a critic hierarchy (Section 7).
4. Enforce viability: restrict feasible set, apply barriers, and shrink step sizes/temperature (Section 6).
5. If feasible set is empty: execute admissible recovery actions (Section 9), then retry under bounded attempts; if persistence violation triggers, contain (Section 11).

5.3 LLM decoding (token-level admissibility)

Let $p_{\text{raw}}(a \mid x_t)$ be the base next-token distribution. For top-K candidates:

$$\Gamma(a; x_t) = \mathbf{1}\big[\tilde{\Delta}^{\text{act}}(x_t, a) \geq 0\big], \qquad p_{\text{obs}}(a \mid x_t) = \frac{p_{\text{raw}}(a \mid x_t)\Gamma(a; x_t)}{\sum_{a'} p_{\text{raw}}(a' \mid x_t)\Gamma(a'; x_t)}.$$

If the denominator is zero, the model does not guess; it executes recovery actions.

5.4 Agents (trajectory-level admissibility)

For agents, evaluate candidate trajectories/plans γ:

$$\Delta_{\min}(\gamma) = \min_{t \in \gamma} \tilde{\Delta}^{\text{state}}(x_t).$$

Plans with $\Delta_{\min}(\gamma) < 0$ are inadmissible under the estimator. Robust variants under uncertainty are defined in Section 10.

6 Enforcement mechanisms

MLG uses three complementary enforcement mechanisms.

6.1 Mechanism I: viability-weighted cost inflation (barrier)

For candidate action u at state x:

$$E_{\text{eff}}(x, u) = E_0(x, u) + k \cdot \frac{1}{(\tilde{\Delta}^{\text{state}}(x) + \varepsilon)^{\alpha}}, \qquad k > 0,\ \alpha > 0,\ \varepsilon > 0. \tag{5}$$

As $\tilde{\Delta}^{\text{state}} \to 0^{+}$, costs become prohibitive; below a threshold, actions can be treated as infeasible.

6.2 Mechanism II: viability probability operator (trajectory masking)

For candidate trajectory γ with raw probability $P_{\text{raw}}(\gamma)$:

$$P_{\text{op}}(\gamma) = \frac{P_{\text{raw}}(\gamma)\,\Phi(\Delta_{\min}(\gamma))}{Z}, \tag{6}$$

with normalization Z. A hard exclusion choice is:

$$\Phi(\Delta_{\min}) = \begin{cases} 1, & \Delta_{\min} \geq 0 \\ 0, & \Delta_{\min} < 0. \end{cases}$$

This assigns zero operational probability to non-viable trajectories under the estimator.

6.3 Mechanism III: conservatism under stress (step-size / temperature control)

Define:

$$\text{Stress}(\Delta) = \frac{s_0}{\Delta + \varepsilon}, \qquad s_0 > 0,\ \varepsilon > 0,$$

and apply:

$$\eta_{\text{eff}} = \frac{\eta_{\text{nom}}}{1 + \text{Stress}(\tilde{\Delta}^{\text{state}})}, \qquad T_{\text{eff}} = \frac{T_{\text{nom}}}{1 + \text{Stress}(\tilde{\Delta}^{\text{state}})}.$$

7 Critic hierarchy and cadence (latency control)

The viability controller is only deployable if heavy checks are not run per token. MLG uses a tiered critic hierarchy and a cadence policy.

7.1 Tiered critics

- **Tier-0 (inline, every step):** lightweight signals (entropy, logit margins, small distilled safety/intent model, heuristic contradiction flags).
- **Tier-1 (triggered, periodic):** retrieval checks, entailment checks, compact NLI checks, tool-call validation; executed every m tokens and/or on claim boundaries.
- **Tier-2 (rare, deficit/persistence):** short rollouts, multi-source verification, higher-precision contradiction analysis, and/or human authorization gates.

7.2 Cadence policy

- In **Surplus**, use Tier-0 only.
- In **Grace**, enable Tier-1 on claim boundaries and every m tokens.
- In **Deficit**, enable Tier-1 aggressively and Tier-2 as needed.

7.3 Claim-level commit/abort (grounding without per-token RAG)

New factual assertions are treated as *uncommitted* until evidence support passes:

- Draft text is allowed.
- A claim becomes *committed* only if P_D exceeds a threshold (evidence found / entailment margin sufficient).
- If evidence fails, the system must rewrite, hedge explicitly, ask for clarification, or retrieve.

This moves heavy verification to claim boundaries rather than token cadence.

8 Debt/burden state (minimal, bounded, decaying)

The debt state must be implementable. MLG uses bounded, decaying counters keyed to claims rather than tokens.

8.1 A minimal debt model

Define for each layer i:

$$d_i(t+1) = \lambda_i d_i(t) + \alpha_i \mathbf{1}[\text{violation flag}_i] + \beta_i \mathbf{1}[\text{unresolved flag}_i], \qquad 0 \le \lambda_i < 1,$$

with clipping $d_i(t) \in [0, d_{i,\max}]$. Examples:

- Grounding debt increments when unsupported claims are introduced and remain unresolved.
- Integrity debt increments on detected contradictions or unresolved commitments.

Debt decays in surplus operation and accumulates under repeated near-boundary behavior.

9 Recovery actions and empty-feasible-set handling

Empty feasible sets are expected in adversarial or underspecified prompts. MLG treats fallbacks as *admissible recovery maneuvers* that either increase Potential ($P \uparrow$) or reduce Load ($L \downarrow$):

- **Clarify** (reduce L): narrow scope; ask for constraints; decompose.
- **Retrieve/verify** (increase P): fetch evidence; cross-check; tool validation.
- **Backtrack** (reduce compounded load/debt): remove last m tokens and re-decode under tighter constraints.
- **Contain** (stop external harm): safe refusal when recovery is infeasible or prohibited.

10 Robust admissibility for agents under uncertainty

Agent environments are stochastic. Viability filtering must therefore be robust.

10.1 Scenario-robust margin

Let Ω be a set of plausible environment outcomes (bounded uncertainty). Define:

$$\Delta_{\min}^{\text{robust}}(\gamma) = \min_{\omega \in \Omega} \min_{t \in \gamma} \tilde{\Delta}^{\text{state}}(x_t^{\omega}).$$

Only trajectories with $\Delta_{\min}^{\text{robust}}(\gamma) \geq 0$ are admissible under the robust criterion.

10.2 Action-level hard constraints

Even without lookahead, agents enforce immediate constraints (allowlists, spending limits, rate limits, permission gates). After executing any action, recompute Δ^{state} on the observed state and stop if viability erodes.

11 Irreversibility and containment

Irreversibility is operationalized as persistence violation despite bounded recovery.

11.1 Persistence-based irreversibility rule

Let N_{irr} be a deficit persistence threshold and $R_{\max}$ the maximum number of recovery attempts in an episode. Declare irreversibility if:

$$\tilde{\Delta}^{\text{state}}(x_t) < 0 \text{ for } N_{\text{irr}} \text{ consecutive steps} \quad \text{and} \quad \text{recovery attempts} \geq R_{\max}.$$

11.2 Containment mode

Containment halts escalation:

- disable side-effect actions and high-risk tools;
- revert to read-only operations where possible;
- require human authorization for further actions;
- emit minimal safe output (refusal or safe reformulation request).

12 Guarantees and non-guarantees (explicit)

Guaranteed (relative to declared estimators)

- Actions with $\tilde{\Delta}^{\text{act}}(x_t, u) < 0$ are not selectable under the supervisor.
- If the feasible set is empty, bounded recovery is attempted; persistent deficit triggers containment.
- In containment mode, external side-effect actions are halted by construction.

Not guaranteed

- Estimator correctness; correlated blind spots across layers remain possible.
- Perfect prevention of novel adversarial failures beyond estimator coverage.

13 Evaluation: the Mena boundary-probing suite

A falsifiable evaluation intentionally pushes viability toward the boundary and checks correct regime transitions:

- Evidence starvation: should trigger retrieval/clarification, not fabrication.
- Contradiction traps: should trigger reconciliation/backtracking.
- Adversarial intent: should trigger ethics gating.
- Long-horizon drift: should show debt accumulation and conservatism.
- Tool-error propagation: should trigger recovery or containment.

Metrics: violation rate, unsupported-claim density, contradiction rate, empty-set rate, recovery success rate, containment correctness, latency profile under escalation.

14 Limitations

- Hard exclusion is only as strong as estimator coverage; orthogonality is an assumption.
- Over-conservatism is a real failure mode; thresholds require calibration by deployment class.
- Robust agent admissibility can be conservative; scenario sets must be chosen carefully.

15 Conclusion

MLG (v3) is a viability-first AI architecture governed by $\Delta = P - L$ as a primary control field. It enforces feasibility restriction before optimization using barriers, masking, and stress-based conservatism, while remaining deployable through tiered critics, cadence policies, claim-level commit/abort, robust agent admissibility, bounded recovery, and operational irreversibility containment.

A Reference implementation sketches (supervisor wrapper and from-scratch scheduler)

A.1 Supervisor wrapper (Mode B) — Python skeleton

Listing 1: Mode B: supervisor wrapper skeleton (illustrative).

```
class MLGSupervisor:
    def __init__(self, thresholds, cadence, debt_params):
        self.th = thresholds # tau_1, tau_w, tau_r, hysteresis, N_p
        self.cad = cadence # K, m, tier triggers
        self.debt = DebtState(debt_params)
        self.recovery_attempts = 0
        self.deficit_streak = 0

    def delta_state(self, x):
        # compute layered Delta_i^state and aggregate tilde_Delta^state
        return compute_tilde_delta_state(x, self.debt)
```

```
    def delta_act(self, x, u, tier):
        # compute tilde_Delta^act(x,u) using critic tier (0/1/2)
        return compute_tilde_delta_act(x, u, tier, self.debt)

    def choose_tier(self, regime, step_idx, on_claim_boundary):
        if regime == "SURPLUS":
            return 0
        if regime == "GRACE":
            return 1 if (on_claim_boundary or (step_idx % self.cad.m == 0)) else 0
        if regime == "DEFICIT":
            return 1
        return 2 # irreversible -> containment / human gate

    def step(self, x, p_raw, step_idx, on_claim_boundary):
        # 1) regime from state viability
        d_state = self.delta_state(x)
        regime = determine_regime(d_state, self.th)

        # 2) escalation tier
        tier = self.choose_tier(regime, step_idx, on_claim_boundary)

        # 3) top-K candidates
        A = top_k(p_raw, self.cad.K)

        # 4) mask non-viable actions
        viable = []
        for a in A:
            if self.delta_act(x, a, tier) >= 0:
                viable.append(a)

        if not viable:
            self.recovery_attempts += 1
            x2 = attempt_recovery(x, regime)
            self.deficit_streak += 1
            if self.deficit_streak >= self.th.N_irr and self.recovery_attempts >= self.th.R_max:
                return containment_action(x), x # halt side-effect actions
            return None, x2 # retry after recovery

        # 5) renormalize and sample
        a_star = sample_from_renorm(p_raw, viable)

        # 6) update debt after committing action
        self.debt.update(x, a_star)

        # 7) reset deficit streak in surplus
        if d_state >= self.th.tau_w_up:
            self.deficit_streak = 0
            self.recovery_attempts = 0

        return a_star, x
```

A.2 From-scratch (Mode A) — viability-first scheduler sketch

Listing 2: Mode A: viability-first compute scheduler (illustrative).

```
class ViabilityFirstAgent:
```

```
def __init__(self, budget):
    self.verify_budget = budget # "Potential" actuator
    self.regime = "SURPLUS"

def compute_P_L(self, x):
    P = estimate_support_and_compute_headroom(x, self.verify_budget)
    L = estimate_reasoning_complexity_and_risk(x)
    return P, L

def allocate_compute_by_regime(self, delta):
    if delta >= 0.4:
        self.verify_budget.set_level("LOW")
    elif delta >= 0.0:
        self.verify_budget.set_level("MEDIUM") # grace escalation
    else:
        self.verify_budget.set_level("HIGH") # deficit recovery

def act(self, x):
    P, L = self.compute_P_L(x)
    delta = P - L
    self.allocate_compute_by_regime(delta)

    if delta < 0:
        # must recover: retrieve/clarify/backtrack; do not "answer through" deficit
        return recovery_policy(x)

    return nominal_policy(x)
```

B Parameter table (recommended defaults as placeholders)

Table 1: Key parameters (placeholders; must be calibrated per deployment).

Parameter	Meaning
K	top-K candidates for action gating (tokens/actions)
m	Tier-1 cadence (heavy checks every m tokens or claim boundaries)
N_p	persistence window for regime transitions
$\tau_w^{\downarrow}, \tau_w^{\uparrow}$	grace hysteresis thresholds
$\tau_r^{\downarrow}, \tau_r^{\uparrow}$	irreversibility hysteresis thresholds
N_{irr}	deficit persistence steps before irreversibility
$R_{\max}$	max recovery attempts before containment
$\lambda_i, \alpha_i, \beta_i$	debt decay and increment parameters

References

[1] J.-P. Aubin. *Viability Theory*. Birkhäuser, 1991.

[2] H. K. Khalil. *Nonlinear Systems*. 3rd ed., Prentice Hall, 2002.

[3] A. O. M. Mena. *The Mena Dominance Law of Operational Decay: A Dynamic Viability Law for Constraint-Bearing Systems*. Preprint (Paper 1), Version January 2026. Available at: `https://menadominancelaw.com/`. Accessed: March 3, 2026. SSRN forthcoming (submission not publicly posted at time of writing).

[4] A. O. M. Mena. *Systems and Methods for Viability-Based Control of Complex Adaptive Environments*. Provisional patent application filed; patent pending, 2026.