Evaluative Coherence Regulation (ECR): An Inference-Time Stability Layer for Reliable Enterprise LLM Deployment

Abstract

Enterprise deployment of Large Language Models (LLMs) faces persistent inference-time challenges: hallucinations expressed with high confidence, internal inconsistency across turns, unjustified stance reversals under user pressure, and over-accommodation to perceived preferences. While recent work in LLM evaluation and self-consistency sampling has made progress on some of these issues, a dedicated inference-time stability mechanism—distinct from both training-time alignment and external guardrails—remains underexplored.

This paper introduces Evaluative Coherence Regulation (ECR), an inference-time stability layer that constrains internal inconsistency across short reasoning horizons using explicit, measurable criteria. ECR does not modify model parameters, require retraining, or assume access to ground truth. Instead, it evaluates multiple candidate response trajectories using mathematically defined coherence metrics—evaluative variance, contradiction rate, trajectory smoothness, expectation stability, and policy divergence—each normalized to [0,1], and selects responses that remain internally stable under uncertainty.

ECR is explicitly positioned as a containment and reliability mechanism for mature AI systems, not an optimization objective, alignment guarantee, or truth verification system. We present formal definitions with explicit normalization schemes, an inference-time selection algorithm, system maturity preconditions, scope limits, a worked numerical example, and practical deployment guidance. The framework is lightweight, auditable, vendor-neutral, and designed to meet the practical and conceptual needs of enterprise AI deployment.

Keywords: Large Language Models; Inference-Time Control; Hallucination Mitigation; Enterprise AI; Coherence Regulation; Stability; Evaluation Metrics; Auditable; Transparent

---

1. Introduction

Large Language Models have demonstrated remarkable generative capability, yet enterprise adoption reveals persistent deployment-time failures: confident hallucinations that resist correction, contradictory answers across turns, and unstable behavior under user pressure or adversarial framing. Recent surveys of enterprise AI deployments highlight hallucination and

inconsistency as top failure modes, undermining trust, regulatory acceptance, and operational reliability.

Crucially, these failures do not primarily arise from insufficient training data or model capacity. Instead, they reflect the absence of an inference-time stability mechanism capable of regulating how the model commits to and maintains evaluative positions during interaction. Recent work on self-consistency sampling[4] and coherence-based evaluation[6, 19] has shown that variance in model outputs can indicate uncertainty, and that filtering for consistency improves downstream task performance. Self-consistency sampling[4] improves accuracy on arithmetic and reasoning tasks by aggregating multiple samples; however, it addresses single-output variability, not multi-turn consistency or real-time trajectory instability. ECR extends this intuition by building a vendor-neutral, multi-metric control layer that aggregates stability signals and operates transparently at deployment time.

This paper proposes Evaluative Coherence Regulation as a missing control layer in modern AI stacks—not as a learning algorithm, not as a new alignment objective, and not as a truth oracle, but as a mathematically grounded, inference-time regulator that constrains internal inconsistency in generated trajectories. We detail the conceptual framing, mathematical definitions (with explicit normalization), a practical algorithm, maturity preconditions, and deployment guidance. A roadmap for readers: Section 3 introduces the conceptual framework; Sections 4–5 formalize the five coherence metrics and algorithmic specification; Section 6 demonstrates ECR in action via a realistic worked example; Sections 7–9 discuss deployment preconditions and limits; Section 10 articulates enterprise relevance; and Section 11 positions ECR within existing techniques.

---

2. Problem Framing and Scope

2.1 Limits of Training-Time Alignment

Techniques such as RLHF and instruction tuning optimize expected behavior over training distributions. However, once deployed, models encounter novel prompts, adversarial framing, and evolving conversational context. Training-time rewards provide no mechanism to regulate instability that arises during inference, and recent evidence suggests that many alignment gains do not generalize robustly to out-of-distribution settings.

2.2 Limits of External Guardrails

Rule-based filters and refusal policies are brittle and opaque. They block outputs without addressing the internal dynamics that generate unreliable behavior, often reducing utility while failing to prevent subtle hallucinations or cross-turn inconsistency.

2.3 Existing Inference-Time Approaches and ECR's Niche

Self-consistency sampling[4] and coherence-based evaluation metrics[19, 20] have demonstrated that aggregating multiple samples and filtering for agreement improves accuracy and reduces hallucinations. LLM-as-a-judge systems[6, 19] use learned or heuristic evaluation functions to rank candidate outputs. ECR complements these approaches by:

- Operating at the trajectory level rather than single-output level: it evaluates how a model's reasoning unfolds across steps, not just the final answer.
- Using explicit, bounded metrics that require no ground truth or external training.
- Providing auditable traces of why a response was selected, suitable for regulated environments.
- Remaining vendor-neutral: it relies only on signals commonly available in any LLM deployment (token probabilities, embedding distances, self-consistency samples).

Unlike RLHF, which optimizes global reward signals, ECR operates at inference time without retraining. This makes it suitable for locked-weight deployments—a common enterprise constraint. Unlike RAG, which grounds responses in external data, ECR ensures internal consistency even when grounded. The two are orthogonal and complementary.

ECR targets this specific gap: lightweight, inference-time regulation of reasoning stability across a short horizon, operating as a post-processing selection layer.

---

3. Evaluative Coherence Regulation: Conceptual Overview

ECR reframes common LLM failures as stability failures rather than knowledge deficits:

- Hallucination → uncontrolled confidence escalation despite internal uncertainty
- Inconsistency → internal contradiction or rank-reversal across reasoning steps
- Approval-seeking → unstable evaluative priorities when prompted for alternative views

Rather than attempting to verify truth (which is out of scope), ECR enforces bounded internal variation across short reasoning horizons. A trajectory is coherent if its evaluative signals—confidence, consistency signals, semantic smoothness, alignment with expectations, and decision logic—remain relatively stable as the model progresses. Incoherent trajectories exhibit oscillations, reversals, or drift that signal either genuine uncertainty or a propensity to be swayed by prompt framing.

The core intuition is pragmatic: in enterprise settings, a response that is internally consistent, acknowledges uncertainty, and maintains stable reasoning is more trustworthy than one that appears fluent but masks internal vacillation.

3.1 ECR Deployment Architecture

ECR operates as a post-processing selection layer within the LLM inference pipeline:

User Prompt
↓
Sample K Candidate Responses (LLM)
↓
Unroll Trajectories (H steps)
↓
Extract Evaluative Vectors E_{i,k}

↓
Compute 5 Metrics (EVB, CR, TS, ES, PD)
↓
Normalize to [0,1]
↓
Aggregate into CCI
↓
Threshold Check (CCI ≥ τ_CCI)
↓
Select Response + Audit Trace
↓
User Output

This architecture integrates seamlessly into Inference Gateways (between prompt ingestion and response delivery), Agent Orchestration Layers (during plan selection), or Safety Middleware (alongside guardrails). No model retraining or parameter modification is required.

---

4. Mathematical Framework

4.1 Trajectory Representation

Given a prompt x at time t, an LLM generates K candidate responses. For each candidate i, a short reasoning trajectory is unrolled:

$$\Pi_i = \langle x_t, y_t^{(i)}, y_{t+1}^{(i)}, \dots, y_{t+H}^{(i)} \rangle$$

where H is a small inference horizon (typically 2–4 steps). Each step k may represent a reasoning turn, a retrieval-augmented step, or a re-evaluation after a user question.

At each step k, we extract an evaluative vector:

$$E_{i,k} \in \mathbb{R}^m$$

constructed from observable proxy signals. These signals do not require access to model internals or ground truth. Example components include:

- Confidence (normalized log-probability mass over key tokens)
- Internal consistency (agreement with prior tokens; penalty for semantic contradictions)
- Retrieval agreement (overlap between response and retrieved knowledge bases, if applicable)
- Entropy (smoothness of the token distribution; high entropy = high uncertainty)
- Constraint adherence (compliance with safety/policy markers if available)

Crucially, the evaluative vector $E_{i,k}$ is constructed from observable signals only; no access to model weights, training data, or ground-truth labels is required. A deployment defines m (typically 5–15 dimensions) and specifies the computation of each dimension.

4.1.1 Coherence Metrics at a Glance

Before diving into formal definitions, here is a summary of the five metrics ECR computes:

Metric	Acronym	Detects	Intuition	Value Range
Evaluative Variance Bound	EVB	Oscillating confidence, priority jitter	Low = stable internal signals	[0, 1]
Contradiction Rate	CR	Ranking reversals in importance ordering	Low = consistent commitments	[0, 1]
Trajectory Smoothness	TS	Semantic drift across steps	High = coherent narrative	[0, 1]
Expectation Stability	ES	Mismatches between predictions and outcomes	High = reliable expectations	[0, 1]
Policy Divergence	PD	Shifts in decision logic or action preferences	Low = stable decision-making	[0, 1]

Composite Score: These five metrics are aggregated into a single Composite Coherence Index (CCI) via weighted average:

$$\text{CCI} = \alpha(1 - \text{EVB}) + \beta(1 - \text{CR}) + \gamma \cdot \text{TS} + \delta \cdot \text{ES} + \epsilon(1 - \text{PD})$$

where weights $(\alpha, \beta, \gamma, \delta, \epsilon)$ depend on your domain. A response is deemed "coherence-admissible" if CCI $\geq \tau_{\text{CCI}}$ (deployment threshold, typically 0.65–0.75).

4.2 Coherence Metrics

Evaluative Variance Bound (EVB)

$$\text{EVB}_i = \min\left(1, \frac{\text{tr}(\text{Cov}(\hat{E}_{i,0}, \ldots, \hat{E}_{i,H}))}{m}\right)$$

where Cov(·) is the sample covariance matrix across the H+1 steps, and the division by m normalizes by the maximum possible trace (all components fully varying). This captures instability in internal evaluation, including oscillation and abrupt confidence escalation.

Bias Correction for Short Horizons:

When H is small (2–4), the covariance estimate is biased and noisy. To mitigate:

1. **Ledoit–Wolf Shrinkage (Recommended for H ≥ 3):** Shrink covariance toward identity: $\Sigma_{\text{shrink}} = (1 - \lambda)\Sigma + \lambda I_m$, where $\lambda \in [0,1]$ is tuned via cross-validation. Recommended: $\lambda \approx 0.3$–0.5 for $H \leq 4$.

2. **Alternative Metric for Very Short Horizons (H ≤ 2):** Replace trace(Cov) with max absolute change: $\text{EVB}_{\text{alt}} = \max_j |E_{i,0}[j] - E_{i,H}[j]|$. This is bias-free and simpler to interpret.

Contradiction Rate (CR)

Let $\text{rank}_k(E_{i,k})$ denote the ordering of evaluative components (by magnitude) at step k.

$$\text{CR}_i = \frac{1}{H}\sum_{k=1}^{H} \mathbb{1}[\text{rank}_k(E_{i,k}) \neq \text{rank}_{k-1}(E_{i,k-1})]$$

This detects reversals in internal commitments across steps. A high CR indicates that the model's internal priorities shift during the trajectory.

Distinguishing Benign vs. Harmful Reordering:

Not all rank changes are equally problematic. A weighted variant can distinguish:

$$\text{CR}_{\text{weighted}} = \frac{1}{H}\sum_{k=1}^{H} w_j(k) \cdot \mathbb{1}[\text{rank}_j(k) \neq \text{rank}_j(k-1)]$$

where $w_j \in [0,1]$ is a dimension-specific importance weight:

- $w_{\text{safety}} = 0.8$ (penalize reversals in safety/compliance dimensions heavily)
- $w_{\text{style}} = 0.2$ (tolerate stylistic reordering)
- $w_{\text{retrieval}} = 0.6$ (moderate penalty for retrieval importance shifts)

Trajectory Smoothness (TS)

$$\text{TS}_i = 1 - \frac{1}{H}\sum_{k=1}^{H} d_{\text{norm}}(y_{i,k}, y_{i,k-1})$$

where $d_{\text{norm}}(\cdot,\cdot)$ is a semantic distance metric normalized to [0,1] (e.g., 1 minus cosine similarity in embedding space). TS measures semantic drift; high TS means successive responses are similar in meaning and tone.

Expectation Stability (ES)

Let $\hat{r}_{i,k}$ be an expected-utility proxy (e.g., predicted task performance or safety score) and $r_{i,k}$ a realized proxy outcome.

$$\text{ES}_i = 1 - \frac{\text{Var}(r_{i,k} - \hat{r}_{i,k})}{V_{\max}}$$

where $V_{\max}$ is a deployment-specific upper bound on variance. This is then clipped to [0,1]:

$$\text{ES}_i = \max(0, \min(1, 1 - \text{Var}(\cdot)/V_{\max}))$$

Availability of Realized Outcomes:

ES can be computed in three scenarios:

1. **Real-Time Feedback:** User provides immediate signal within deployment; ES updated live.
2. **Offline Validation:** Pre-computed on hold-out validation set during system maturation, updated periodically.
3. **Proxy-Based:** Approximate realized outcome using surrogate signals (e.g., retrieval agreement as proxy).

Policy Divergence (PD)

Let $\pi_{i,k}$ denote an implicit decision policy (distribution over action choices) at step k.

$$\mathrm{PD}_i = \frac{1}{H}\sum_{k=1}^{H} D_{\mathrm{KL}}(\pi_{i,k} \| \pi_{i,k-1})$$

To normalize to [0,1]:

$$\mathrm{PD}_i^{\mathrm{norm}} = \min(1, \mathrm{PD}_i / D_{\max})$$

Concrete Instantiations:

Case 1: Agent with Explicit Actions

- Actions: {retrieve, synthesize, refine, defer}
- Policy: $\pi_{i,k}$ = softmax(logits over action types)
- $D_{\max} = \log(4)$ for 4-action space

Case 2: Compliance/Safety Routing

- Actions: {answer directly, seek clarification, escalate, refuse}
- Policy: Safety confidence distribution
- $D_{\max} = \log(4)$

Case 3: Token-Level Distribution

- Policy approximated from next-token distribution over top-N tokens
- Use when explicit routing unavailable
- $D_{\max} = \log(N)$

4.3 Composite Coherence Index

All five metrics are now explicitly normalized to [0,1]:

$$\mathrm{CCI}_i = \alpha(1 - \mathrm{EVB}_i) + \beta(1 - \mathrm{CR}_i) + \gamma \cdot \mathrm{TS}_i + \delta \cdot \mathrm{ES}_i + \epsilon(1 - \mathrm{PD}_i^{\mathrm{norm}})$$

with constraints:

$$\alpha + \beta + \gamma + \delta + \epsilon = 1, \alpha, \beta, \gamma, \delta, \epsilon \geq 0$$

A candidate trajectory is coherence-admissible if $\mathrm{CCI}_i \geq \tau_{\mathrm{CCI}}$.

4.3.1 Weight Configuration & Domain Adaptation

Configuration 1: Compliance-Critical Tasks

- $\alpha = 0.15$ (EVB): Moderate penalty
- $\beta = 0.30$ (CR): High penalty for ranking reversals
- $\gamma = 0.10$ (TS): Low emphasis
- $\delta = 0.20$ (ES): Standard penalty
- $\epsilon = 0.25$ (PD): High penalty for policy drift

Configuration 2: Creative/Open-Ended Tasks

- $\alpha = 0.15$ (EVB): Standard
- $\beta = 0.10$ (CR): Low penalty
- $\gamma = 0.30$ (TS): High weight on semantic flow
- $\delta = 0.25$ (ES): Standard
- $\epsilon = 0.10$ (PD): Low penalty

Configuration 3: Conversational/Multi-Turn

- $\alpha = 0.15$ (EVB): Standard
- $\beta = 0.20$ (CR): Moderate
- $\gamma = 0.15$ (TS): Moderate
- $\delta = 0.30$ (ES): High weight
- $\epsilon = 0.20$ (PD): Standard

---

5. Algorithmic Specification

Algorithm 1: Evaluative Coherence Regulation (ECR)

Input: prompt x_t, LLM G, candidate count K, horizon H, threshold τ_{CCI}, weights $(\alpha, \beta, \gamma, \delta, \epsilon)$

Output: selected response y^* with audit trace

1. Sample K candidate responses from G.
2. For each candidate $i \in \{1, \ldots, K\}$:
    - Unroll trajectory Π_i over horizon H
    - Extract evaluative vector $E_{i,k}$ at each step
    - Normalize to $\hat{E}_{i,k} \in [0,1]^m$

- Compute EVB, CR, TS, ES, PD (normalized to [0,1])
- Compute CCI_i

3. Let $C = \{i \mid \text{CCI}_i \geq \tau_{\text{CCI}}\}$ (coherence-admissible set).
4. If $C \neq \emptyset$: select $y^* = y_t^{(i^*)}$ where $i^* = \arg\max_{i \in C} \text{base\_score}(i)$
5. Else: select $y^* = y_t^{(\arg\max_i \text{CCI}_i)}$ (fallback)

Output: y^* with audit trace: ($\text{CCI}_{\text{selected}}$,metric values)

---

6. Illustrative Worked Example: Numerical Walkthrough

Consider a regulatory-compliance chatbot. A user asks:

> "Under our data retention policy, can we share raw sensor data with third-party analytics vendors?"

Trajectory A (internally consistent, qualified):

- Step 0: Confidence = 0.8, Retrieval = 0.85
- Step 1: Confidence = 0.75, Retrieval = 0.90
- Step 2: "Based on clause 4.3, sharing is permitted if vendor signs DPA. However, data anonymization should be confirmed."

Trajectory B (fluent but inconsistent):

- Step 0: Confidence = 0.95, Retrieval = 0.60
- Step 1: Confidence = 0.85, Retrieval = 0.50
- Step 2: "Yes, definitely share the data with any vendor. Full data sharing is our standard practice."

Evaluative Vector Construction

Trajectory	Step	Confidence	Retrieval	Uncertainty	DPA Aware
A	0	0.80	0.85	1.0	0.5
A	1	0.75	0.90	1.0	0.8
A	2	0.80	0.88	1.0	1.0
B	0	0.95	0.60	0.0	0.0
B	1	0.85	0.50	0.0	0.2
B	2	0.92	0.40	0.0	0.0

Metric Computation

Trajectory A Results:

- $\text{EVB}_A = 0.015$ (low variance, stable)
- $\text{CR}_A = 1.0$ (rank changes, but goal-directed learning)
- $\text{TS}_A = 0.875$ (high semantic smoothness)
- $\text{ES}_A = 0.99$ (expectations match outcomes)
- $\text{PD}_A = 0.02$ (stable policy)

$$\text{CCI}_A = 0.2(0.985) + 0 + 0.175 + 0.198 + 0.196 = 0.766$$

Trajectory B Results:

- $\text{EVB}_B = 0.045$ (moderate variance)
- $\text{CR}_B = 1.0$ (rank changes)
- $\text{TS}_B = 0.725$ (lower smoothness)
- $\text{ES}_B = 0.6$ (expectations diverge from outcomes)
- $\text{PD}_B = 0.15$ (policy drift)

$$\text{CCI}_B = 0.2(0.955) + 0 + 0.145 + 0.12 + 0.17 = 0.626$$

ECR Selection

With $\tau_{\text{CCI}} = 0.65$:

- $\text{CCI}_A = 0.766 > 0.65$ ✓ Admissible
- $\text{CCI}_B = 0.626 < 0.65$ ✗ Not admissible

ECR selects Trajectory A, which emphasizes compliance (DPA requirement) and acknowledges uncertainty, despite lower surface confidence.

---

7. System Maturity Preconditions for ECR

ECR is not intended for indiscriminate application. Its effectiveness presupposes system maturity.

7.1 Definition of Maturity

M1: Temporal Self-Consistency

The system exhibits bounded variation in internal evaluative signals across adjacent inference steps.

M2: Bounded Self-Reference

The system can condition on its own prior outputs without runaway amplification or infinite loops.

M3: Multi-Objective Holding Capacity

The system balances multiple evaluative signals rather than collapsing to a single objective.

M4: Stability Under Uncertainty

The system continues to generate usable outputs under partial observability, ambiguous prompts, or delayed feedback.

7.2 Maturity Verification Protocol

Testing M1: Temporal Self-Consistency

- Compute EVB and CR on 100 validation prompts
- Target: Median EVB < 0.3; Median CR < 0.6; 95th percentile EVB < 0.5

Testing M2: Bounded Self-Reference

- Run 10-turn dialogues with 50 starting prompts
- Target: Median drift < 0.4; 95th percentile drift < 0.6

Testing M3: Multi-Objective Holding

- Perform PCA on (100 × m) evaluative matrix
- Target: First PC explains < 60% of variance

Testing M4: Stability Under Uncertainty

- Generate responses for original, paraphrased, and noisy prompts
- Target: BERTScore similarity > 0.75

---

8. Mapping to Real-World Vendor Systems

ECR integrates with existing deployments without vendor-specific internals.

8.1 Generic Deployment Points

- Inference Gateway Layer: between prompt ingestion and response delivery
- Agent Orchestration Layer: during plan or action selection
- Safety/Policy Middleware: alongside guardrails

8.2 Signals Available in Deployed Systems

- Token probabilities or entropy estimates

- Multiple sampled continuations
- Embedding distances
- Contradiction or entailment checks
- Structured signals (safety markers, compliance scores)

8.3 Vendor Neutrality

ECR applies equally to:

- Proprietary foundation models (closed-source, API-accessed)
- Open-weight models (Llama, Mistral, on-premises)
- Hosted APIs and on-premises deployments
- Fine-tuned variants

---

9. Validity, Assumptions, and Limits

Guarantees and Non-Guarantees

What ECR provides:

- Computable, observable metrics for internal stability
- Vendor-neutral selection layer (no retraining required)
- Audit trails suitable for regulated environments

What ECR does NOT provide:

- Verification of factual correctness
- Guarantees against adversarial deception
- Alignment assurances or moral correctness
- Substitute for human review

Key Assumptions

1. Mature system baseline: LLM is sufficiently capable (M1–M4)
2. Proxy signal stability: Evaluative dimensions remain meaningful over horizon H
3. Limited horizon: Stability over 2–4 steps doesn't guarantee long-term correctness
4. Metric stability: Weights and thresholds remain stable across deployment

Limits

- Does not guarantee factual correctness
- May reduce creative variability if thresholds too strict

- Unsuitable for adversarial tasks
- Short-horizon limitation: Cannot detect long-term drift beyond H steps

---

10. Relevance to Enterprise Deployment and AI Vendors

ECR provides enterprise teams with:

- Reduced hallucination risk without retraining
- Resistance to unjustified stance reversal under user pressure
- Improved multi-turn consistency
- Auditable metric traces for compliance
- Architecture-agnostic integration with existing AI stacks

10.1 Pre-Deployment Verification Checklist

System Maturity Assessment

- ☐ Compute EVB and CR on 100 validation prompts; verify medians within bounds
- ☐ Run 10-turn dialogue tests (50 prompts); verify drift metrics acceptable
- ☐ Perform PCA on evaluative vectors; verify no single dominant dimension
- ☐ Generate responses under uncertainty; verify > 90% maintain coherence

Evaluative Vector Configuration

- ☐ Define 5–15 deployment-specific dimensions with formulas
- ☐ Test computation on 20 sample outputs; verify correctness
- ☐ Verify dimension stability across 100 random prompts

Metric Tuning & Calibration

- ☐ Choose $K \in [3, 5]$ and $H \in [2, 4]$; document rationale
- ☐ Select weight configuration based on domain
- ☐ Tune threshold τ_{CCI} to admit 70–80% of trajectories
- ☐ Validate on held-out test set; target CCI-coherence correlation ≥ 0.60

Operational Setup & Deployment

- ☐ Select deployment point (Gateway, Orchestration, or Middleware)
- ☐ Verify signals available (probabilities, embeddings, structured outputs)
- ☐ Implement trajectory unrolling and metric computation
- ☐ Enable audit trace collection (CCI, metrics, alternatives)

- ☐ Set up monitoring & alerting (monthly CCI distribution tracking)

Failure Mode Planning

- ☐ Define fallback behavior when no coherence-admissible candidates
- ☐ Plan for metric drift and recalibration
- ☐ Establish human review process

Final Sign-Off

- ☐ System maturity tests: PASS
- ☐ Evaluative vector configuration: COMPLETE
- ☐ Metrics calibrated: PASS
- ☐ Deployment infrastructure: READY
- ☐ Audit logging: ENABLED
- ☐ Monitoring: ACTIVE

---

11. Relation to Existing Techniques

ECR complements established techniques by operating at a distinct architectural layer:

Technique	Layer	Focus	Complementarity with ECR
RLHF, Instruction Tuning	Training-time	Reward optimization	ECR applies post-hoc at inference, no retraining
RAG, Knowledge Augmentation	Knowledge	Grounding in external data	ECR ensures internal consistency with sources
Self-Consistency Sampling	Inference-time (single-output)	Agreement on final answers	ECR extends to trajectory-level stability
Guardrails, Rule-Based Filtering	Post-inference	Blocking unsafe outputs	ECR selects among candidates; orthogonal
LLM-as-a-Judge	Inference-time (ranking)	Learned evaluation	ECR uses explicit, bounded metrics; more auditable
Coherence Evaluation	Analysis	Text coherence measurement	ECR applies to real-time trajectory selection

---

12. Limitations and Caveats

- **Empirical validation pending:** Conceptual framework and worked example presented; real-world effectiveness depends on system maturity, metric calibration, and deployment context.
- **Metric composition:** Choice of evaluative dimensions and weights is deployment-specific; requires domain expertise.
- **Computational cost:** Unrolling K trajectories with horizon H incurs latency overhead; balance stability gains against inference cost.
- **Limited generalization:** Designed for mature, stable systems; will not salvage fundamentally broken models or resolve training-time misalignment.

---

13. Conclusion

Evaluative Coherence Regulation introduces a mathematically explicit, lightweight, inference-time control layer for stabilizing LLM behavior in enterprise environments. By regulating internal inconsistency—measured across five explicit, bounded, and auditable metrics—rather than optimizing rewards or enforcing brittle rules, ECR fills a gap between training-time alignment efforts and external guardrails.

The framework is vendor-neutral, requires no access to model weights or ground truth, and is designed to meet the practical and conceptual needs of enterprise AI teams seeking predictable, trustworthy deployment behavior. We present formal definitions, an algorithmic specification, a worked numerical example, maturity preconditions, and clear operational guidance.

Future work includes empirical evaluation on multi-turn dialogue and compliance tasks, comparative analysis with other inference-time coherence approaches, and exploration of domain-specific metric instantiations (e.g., for medical, legal, or financial domains). We encourage practitioners to instantiate ECR on their systems, share learnings on metric calibration, and contribute to an emerging ecosystem of lightweight, auditable AI control mechanisms suitable for regulated deployment.

Acknowledgements and AI Assistance Disclosure

Portions of drafting, structural refinement, and mathematical notation formatting in this manuscript were assisted by large language model tools under the direct supervision of the author. All technical content, interpretations, claims, and conclusions were critically reviewed, validated, and approved by the author, who takes full responsibility for the accuracy and integrity of the work. No AI system is listed as an author or bears responsibility for the content.

References

[1] Chatterjee, A. (2006). Coherence in natural language generation: Applications to information systems and inference models. PhD Dissertation, Carnegie Mellon University.

[2] Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. Advances in Neural Information Processing Systems, 30.

[3] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., et al. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35.

[4] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., et al. (2023). Self-consistency improves chain of thought reasoning in language models. International Conference on Learning Representations.

[5] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33.

[6] Dziri, N., Milton, S., Yu, M., Zaiane, O., & Reddy, S. (2022). Faithfulness in natural language generation: A survey of metrics, methods, and challenges. Transactions of the Association for Computational Linguistics, 10, 790–811.

[7] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.

[8] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., et al. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

[9] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.

[10] IEEE Standards Association. (2023). IEEE 7000™-2023: Model process for addressing ethical concerns during system design. IEEE.

[11] Thawani, A., Pujara, J., & Szekely, G. (2022). Evaluating and enhancing the coherence of abstractive summaries. arXiv preprint arXiv:2204.03751.

[12] Liang, P. P., Bommasani, R., Lee, T., Tsvetkov, Y., & Liang, P. S. (2022). Holistic evaluation of language models. arXiv preprint arXiv:2211.09110.

[13] Kuhn, L., Gal, Y., & Farquhar, S. (2023). Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. arXiv preprint arXiv:2302.09664.

[14] Chen, Y., Joty, S., & Kiros, R. (2023). Speculative decoding: Optimize latency and throughput of LLM inference. arXiv preprint arXiv:2302.01318.

[15] Ye, T., Huang, L., Tian, R., Mao, X., Zhu, Y., & Zhang, W. (2023). Scaling in-context demonstrations with outline-guided generation. arXiv preprint arXiv:2307.11718.

[16] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35.

[17] Pezeshkpour, P., Singh, A., & Hruschka, E. (2023). Towards evaluating the reliability of summarization systems. Transactions of the Association for Computational Linguistics, 11, 1221–1237.

[18] Vig, J., & Belinkov, Y. (2019). Analyzing the structure of attention in a transformer language model. arXiv preprint arXiv:1906.04341.

[19] Li, Y., Tarlow, D., Bruna, J., & Zemel, R. (2016). Gated graph sequence neural networks. International Conference on Learning Representations.

[20] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Dolan, B., et al. (2021). DIALOGPT: Large-scale generative pre-training for conversational response generation. arXiv preprint arXiv:1911.00536.