Overtuning by Metaprompt: A Non-Parametric Control Axis for LLM Behavior

Edervaldo Jose de Souza Melo
Independent Researcher
Campo Grande, MS, Brazil
ORCID: 0009-0003-6835-135X
email: edersouzamelo@gmail.com

***Abstract*—Large Language Models (LLMs) are predominantly controlled through parameter-level optimization, with fine-tuning often treated as the primary mechanism for behavioral alignment and specialization. While effective in many contexts, fine-tuning alone operates over globally distributed training data, leaving inference vulnerable to coherence loss when reasoning spans distant or weakly related knowledge regions—an effect frequently associated with hallucination phenomena.**

This paper argues that fine-tuning should not be regarded as the sole legitimate axis of behavioral control in LLM-based systems. We introduce overtuning, a non-parametric control paradigm that governs model behavior externally through an orchestration layer, without modifying model weights. Central to this approach is the concept of a metaprompt: a software-mediated control artifact responsible for dynamically assembling, constraining, and injecting context-aware prompts into one or multiple LLMs based on persistent memory, pre-registered policies, and execution-time conditions.

We present a minimal executable framework in which overtuning reduces effective inferential distance by anchoring generation to dynamically maintained contextual and mnemonic pillars, thereby mitigating hallucination risk through constraint rather than additional learning. Beyond contextual grounding, the framework supports ethical enforcement, role scoping, behavioral routing, and auditability via specialized constraint modules operating at the orchestration level.

At the same time, we explicitly address a critical risk of overtuning: excessive internal coherence may lead to epistemic enclosure, where systems become circularly self-consistent and insufficiently exposed to external reference signals—a phenomenon analogous to Plato's Cave. We argue that this risk does not invalidate overtuning, but instead defines concrete governance requirements, including de-circularization mechanisms, external refresh policies, and formal system-level invariants.

By formalizing overtuning as a distinct control axis and articulating its architectural benefits and limits, this work contributes a governance-oriented alternative to parameter-centric alignment, with direct implications for explainability, auditability, and the design of controllable AI systems.

***Index Terms*—Large Language Models; Overtuning; Metaprompt; Behavioral Control; Non-Parametric Governance; AI Orchestration; Hallucination Mitigation; Explainable AI; Ethical AI by Design; Cognitive System Architecture**

I. Introduction

The rapid adoption of Large Language Models (LLMs) has intensified the search for effective mechanisms to control, specialize, and govern model behavior. To date, fine-tuning—the adjustment of model parameters through additional training—has been widely treated as the dominant, and often exclusive, axis of behavioral control [1], [2]. While fine-tuning remains a powerful technique, its primacy has led to an implicit assumption: that meaningful behavioral alignment must necessarily occur at the level of model weights.

This assumption becomes increasingly fragile in system-level deployments, where LLMs operate as components within larger interactive architectures rather than as isolated predictors. Recent work on prompting strategies and agentic frameworks has highlighted the growing importance of orchestration, tool use, and execution-time control as determinants of model behavior [3]–[5]. In these settings, behavioral outcomes emerge not only from learned parameters, but from the structured interaction between models, prompts, memory, and external control logic.

A critical limitation of parameter-centric control lies in its dependence on globally distributed training data. When inference spans heterogeneous or weakly connected knowledge regions, coherence may degrade, increasing the likelihood of hallucination or speculative generation [6], [7]. This risk is not merely a function of model scale or training quality, but of effective inferential distance between the active reasoning context and the informational anchors available during generation.

This paper challenges the assumption that fine-tuning constitutes the only legitimate axis of behavioral control for LLM-based systems. While prompt engineering has emerged as a practical alternative, it typically relies on static or weakly stateful instruction patterns and lacks formal governance guarantees [7], [8]. We argue that a distinct control paradigm is required—one that operates externally to model parameters and treats behavioral governance as a system-level concern rather than a training artifact.

Central to this approach is the metaprompt, a structured control artifact that mediates between user input, system state, and one or more LLMs. By maintaining persistent contextual anchors and enforcing explicit constraints, overtuning reduces effective inferential distance without expanding the model's underlying knowledge. This enables improved contextual grounding, ethical enforcement, and auditability while preserving model generality.

At the same time, overtuning introduces its own risks. Excessive reliance on internally maintained context and memory may lead to epistemic enclosure, where systems

become increasingly self-consistent but insufficiently exposed to external reference signals. Addressing this tension is not optional: it defines the governance requirements that distinguish robust overtuning architectures from naïve prompt-based control.

The contribution of this paper is threefold: (i) to formalize overtuning as a distinct, non-parametric axis of behavioral control; (ii) to present a minimal executable framework that operationalizes this concept through metaprompt-driven orchestration; and (iii) to articulate the governance mechanisms necessary to balance contextual coherence with epistemic openness. In doing so, we position overtuning as a complementary alternative to fine-tuning, with direct implications for explainability, auditability, and ethical AI system design.

II. Definitions

A. Fine-Tuning

Fine-tuning refers to the modification of an LLM's internal parameters through additional training on task-specific or domain-specific data. Behavioral changes achieved through fine-tuning are embedded within the model weights and persist independently of runtime context. While effective for specialization, fine-tuning is inherently static and opaque, offering limited control granularity once deployment begins.

B. Overtuning

Overtuning is defined as external, non-parametric control of LLM behavior through architectural, orchestration, and policy mechanisms, without modification of model weights. In an overtuned system, behavioral constraints are enforced at runtime via explicit control logic rather than implicit parameter adaptation.

Overtuning operates by restricting, shaping, or routing inference through dynamically maintained contextual and mnemonic anchors. As such, it does not expand model knowledge, but constrains the space of admissible generations. This paradigm emphasizes auditability, reversibility, and governance, positioning behavioral control outside the statistical substrate of the model.

C. Metaprompt

A metaprompt is a software-controlled orchestration artifact responsible for dynamically assembling, transforming, and injecting prompts into one or multiple LLMs based on pre-registered context, persistent memory, and execution-time policies.

Unlike static system prompts, a metaprompt functions as an active control layer. It mediates between user input, stored state, and governance rules, enabling:

- dynamic contextual grounding,
- enforcement of behavioral constraints,
- routing across roles or models, and
- preservation of user-specific state without retraining.

Within an overtuning architecture, the metaprompt constitutes the primary mechanism by which external control is applied to otherwise general-purpose models.

D. Constraint Modules

Constraint modules are specialized functional components operating within the overtuning layer. Each module is responsible for enforcing scoped restrictions, validations, or vetoes over generation behavior, such as ethical boundaries, role limitations, or audit requirements.

These modules are external to the LLM and act deterministically on prompts, context, or outputs. While they may be conceptually represented as personas for design clarity, their role in this framework is strictly architectural: to provide explicit, inspectable points of behavioral governance.

E. Inferential Distance

Inferential distance denotes the effective separation between the active reasoning context and the informational anchors available to the model during generation. Large inferential distances increase the likelihood of speculative inference and hallucination. Overtuning reduces this distance by constraining generation to dynamically refreshed contextual and mnemonic pillars rather than relying solely on latent statistical associations.

III. Minimal Executable Framework for Overtuning

This section presents a Minimal Executable Framework (MEF) that operationalizes overtuning as a non-parametric control axis. The framework is intentionally constrained to the smallest set of components required to demonstrate behavioral governance, auditability, and ethical containment without model retraining.

A. Architectural Overview

The framework adopts a system-oriented view in which the LLM is treated as an encapsulated dependency, not as the locus of governance. Behavioral control is implemented externally through an orchestration layer that coordinates interaction, memory, policy enforcement, and generation.

At a high level, the architecture consists of four core components:

1. Interface Layer — captures user input and delivers system output.

2. Orchestration Layer (Overtuning Layer) — governs execution flow, assembles prompts, and enforces constraints.

3. Memory Layer — maintains persistent and session-level contextual anchors.

4. LLM Adapter — provides controlled access to one or more external LLMs.

This separation ensures that behavioral governance remains explicit, inspectable, and reversible, independent of the statistical substrate of the model.

Rather than documenting implementation details, the use case diagram abstracts the functional roles involved in behavioral governance, emphasizing separation between language model inference and external control actors.

Figure 1 illustrates the system boundaries and the placement of the overtuning layer relative to the LLM.

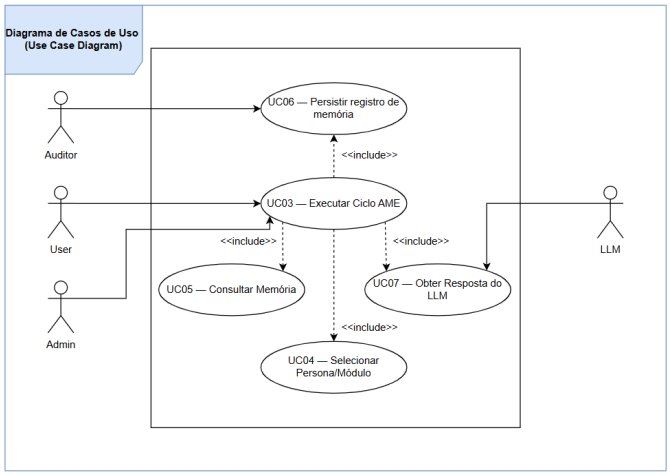

Figure 1. Use Case Diagram illustrating the functional scope of behavioral governance in an overtuned LLM system. The diagram emphasizes the externalization of control, audit, and memory management from the language model.

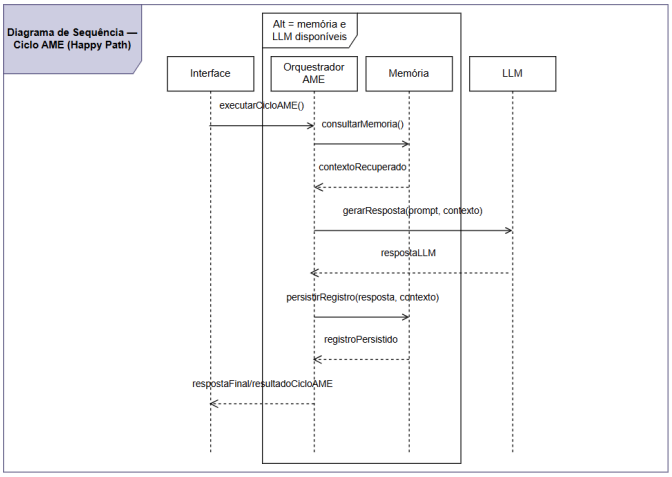

Figure 2. Sequence Diagram of the overtuning execution cycle. The diagram illustrates how metaprompt-driven orchestration constrains inference through dynamic context retrieval, policy enforcement, and validation stages, reducing effective inferential distance without modifying model parameters.

B. Execution Cycle and Control Flow

The minimal execution cycle proceeds as follows:

- User input is received by the Interface Layer.
- The Orchestration Layer initializes a controlled execution context.
- Relevant contextual and mnemonic anchors are retrieved from the Memory Layer.
- The metaprompt dynamically assembles a constrained prompt based on policies, roles, and state.
- The LLM Adapter invokes the selected model with controlled parameters.
- The response is validated, logged, and persisted according to governance rules.
- The final output is returned to the user.

This cycle explicitly decouples generation from governance. The LLM produces candidate responses; the overtuning layer determines whether those responses are admissible within system constraints.

The execution cycle is intentionally represented as a constrained sequence, highlighting validation and policy enforcement stages as first-class elements of inference control.

Figure 2 presents the sequence diagram for the execution cycle, including both normal operation and failure paths.

C. Metaprompt as an Active Control Artifact

Within the framework, the metaprompt functions as an active orchestration artifact, not a static instruction block. Its responsibilities include:

- aggregating user input with retrieved context,
- injecting role and scope directives,
- enforcing ethical and operational constraints,
- adapting prompt structure based on execution state.

These modules embody ethics by design: ethical behavior emerges not from model alignment alone, but from enforceable architectural constraints.

Because the metaprompt is generated and applied at runtime, it enables dynamic behavioral modulation without retraining. This allows overtuning systems to adapt across tasks, domains, and even model providers while preserving continuity of context and governance.

D. Constraint Modules and Ethical Containment

Behavioral containment is enforced through constraint modules operating within the orchestration layer. Each module applies deterministic checks or transformations to prompts, context, or outputs. Examples include: ethical scope enforcement, role and authority validation, tone and risk modulation, audit and logging requirements.

These modules embody ethics by design: ethical behavior emerges not from model alignment alone, but from enforceable architectural constraints. Because constraint modules are external to the LLM, they remain auditable and adjustable without modifying model parameters.

Conceptually, these modules may be represented as personas for design clarity; however, within this framework they are strictly functional governance components.

E. Hallucination Mitigation via Inferential Distance Reduction

Overtuning mitigates hallucination risk by reducing effective inferential distance during generation. Rather than relying solely on latent associations across dispersed training data, the system constrains inference to a dynamically maintained set of contextual and mnemonic anchors.

By continuously injecting relevant context and enforcing scope boundaries, overtuning narrows the admissible generation space. This does not eliminate hallucinations categorically, but structurally reduces their incidence by minimizing speculative leaps across weakly connected knowledge regions.

F. Risk of Epistemic Enclosure and De-Circularization

While overtuning increases local coherence, excessive reliance on internally maintained context introduces the risk of epistemic enclosure. Systems may become circularly self-consistent, progressively detached from external reference signals—an effect analogous to Plato's Cave.

This risk does not invalidate overtuning, but defines its governance requirements. Robust overtuning architectures must therefore incorporate de-circularization mechanisms, such as: periodic refresh from external data sources, bounded temporal persistence of memory, cross-validation against independent models or modules, and human-in-the-loop audit checkpoints.

These mechanisms ensure that contextual coherence does not devolve into epistemic isolation.

G. Deployment Considerations

From a deployment perspective, the framework is agnostic to specific infrastructure choices. The overtuning layer may be implemented as a backend service mediating between client applications, memory stores, and external LLM APIs.

Crucially, the framework assumes no fine-tuning at deployment time. Behavioral control, ethical enforcement, and auditability are achieved entirely through orchestration, enabling rapid iteration and controlled evolution.

The deployment view prioritizes architectural separation, illustrating how governance and memory components remain external to the LLM service while retaining full control over interaction flow.

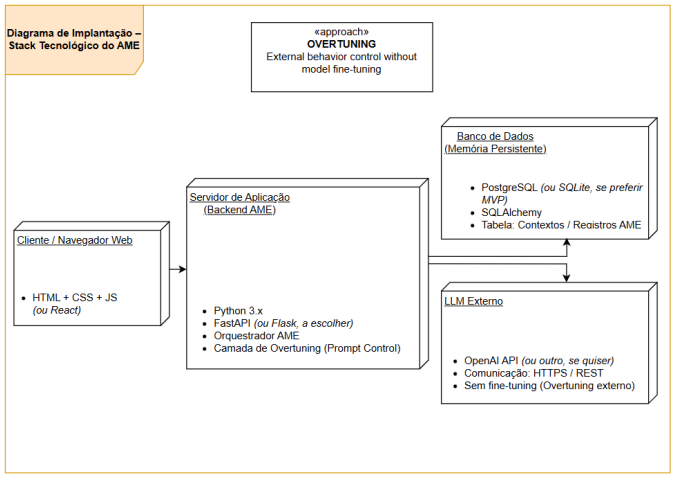

Figure 3. Deployment Diagram of an overtuned LLM system. The figure shows how orchestration, governance, and memory layers operate independently from the external LLM service, enabling behavioral control, ethical enforcement, and auditability without retraining or parameter modification.

Figure 3 illustrates a representative deployment configuration highlighting the separation between orchestration, memory persistence, and external LLM services.

IV. Discussion

A. Overtuning versus Fine-Tuning

Fine-tuning has traditionally been treated as the primary mechanism for aligning and specializing LLM behavior. By embedding task-specific patterns into model parameters, fine-tuning offers durable behavioral change but at the cost of opacity, rigidity, and limited post-deployment control. Once deployed, behavioral deviations often require retraining cycles, complicating auditability and rapid correction.

Overtuning introduces a complementary control axis. Rather than modifying internal representations, it governs behavior through explicit, runtime-enforced constraints. This architectural separation yields three immediate advantages: reversibility, inspectability, and scope precision. Behavioral policies can be adjusted, revoked, or refined without retraining, enabling governance mechanisms that are difficult to achieve through parameter updates alone.

Importantly, overtuning does not aim to replace fine-tuning universally. Instead, it reframes fine-tuning as one tool among others, appropriate for embedding general capabilities or domain priors, while overtuning governs situational behavior, ethical boundaries, and contextual grounding at execution time.

B. Overtuning versus Prompt Engineering

Prompt engineering is often cited as an alternative to fine-tuning. However, prompt engineering typically relies on static or semi-static prompt templates, lacking persistent state, formal governance, or audit guarantees. As such, it remains fragile under complex, long-running interactions.

Overtuning subsumes prompt engineering within a broader architectural paradigm. The metaprompt is not a handcrafted instruction string, but a software-mediated control artifact dynamically generated from system state, memory, and policy. Constraint modules further distinguish overtuning from ad hoc prompting by introducing enforceable boundaries rather than heuristic guidance.

This distinction is critical: overtuning transforms prompting from a craft practice into a governed system capability.

C. Relation to Retrieval-Augmented Generation (RAG)

While overtuning shares surface-level similarities with Retrieval-Augmented Generation (RAG), particularly regarding dynamic context injection, the two approaches address fundamentally different problem scopes.

RAG focuses on augmenting language models with external knowledge retrieval mechanisms to reduce factual hallucination and improve response accuracy. In contrast, overtuning operates as an architectural governance paradigm, enforcing behavioral constraints, role separation, and execution boundaries through non-parametric control mechanisms.

Importantly, overtuning mechanisms are not limited to retrieval-based augmentation and may operate independently of external knowledge sources. Instead, overtuning treats governance, constraint enforcement, and behavioral routing as first-class architectural concerns rather than auxiliary optimization techniques.

D. Governance, Ethics, and Auditability

A central contribution of overtuning lies in its treatment of ethics and governance as architectural concerns rather

than emergent model properties. Ethical constraints are encoded as explicit policies and enforced through deterministic mechanisms external to the LLM.

This approach enables post-hoc inspection, compliance verification, and accountability—capabilities increasingly demanded in regulated or high-stakes domains. By externalizing governance, overtuning aligns behavioral control with established principles of system design, where invariants and constraints are explicitly declared and enforced.

Moreover, governance mechanisms serve not only ethical purposes but epistemic ones. Constraint modules and constitutional policy layers act as safeguards against uncontrolled drift, hallucination amplification, and internal circularity.

E. Limits and Open Challenges

Despite its advantages, overtuning introduces non-trivial challenges. Chief among these is the risk of epistemic enclosure: excessive contextual anchoring may reduce exposure to external reference signals, producing systems that are coherent but increasingly insular.

Additionally, overtuning architectures require careful design to avoid excessive complexity in orchestration logic. Poorly structured constraint layers may introduce unintended interactions or performance overhead.

Finally, while overtuning enhances auditability at the system level, it does not inherently render the internal reasoning processes of LLMs transparent. As such, overtuning should be viewed as necessary but not sufficient for full explainability.

V. Implications for Explainable and Governable AI

The formalization of overtuning has direct implications for Explainable AI (XAI). By shifting behavioral control to an explicit orchestration layer, overtuning creates new loci of explanation: policies, constraints, execution paths, and decision logs become explainable artifacts in their own right.

This reframing suggests a complementary view of explainability. Rather than focusing exclusively on internal model interpretability, systems may achieve practical transparency through governed behavior, traceable decisions, and auditable control flows. In this sense, overtuning provides an architectural bridge between opaque statistical models and human-interpretable governance mechanisms.

VI. Limitations

This work formalizes the overtuning paradigm and presents a minimal executable framework for non-parametric behavioral control of large language models. However, the current contribution is primarily conceptual and architectural in nature.

Empirical validation remains future work. Planned extensions include:

- Implementation of the AME framework to evaluate behavioral compliance and constraint adherence
- Comparative analysis against baseline prompting, fine-tuning, and RAG-based approaches
- Measurement of latency and computational overhead introduced by metaprompt orchestration
- User-centered evaluations of auditability and governance transparency

These investigations aim to complement the architectural contribution presented here with quantitative evidence.

VII. Conclusion and Future Work

This paper has argued that fine-tuning should not be regarded as the sole legitimate axis of behavioral control for LLM-based systems. While parameter adaptation remains effective for embedding general capabilities and domain priors [1], [2], it offers limited flexibility for runtime governance, auditability, and contextual modulation. By externalizing behavioral control through architecture-level orchestration, the overtuning paradigm reframes alignment as a system-level concern rather than a purely training-time artifact.

Beyond flexibility and governance, overtuning contributes to the mitigation of hallucination by reducing effective inferential distance during generation. When behavioral control relies exclusively on parameter adaptation, inference may span weakly connected or heterogeneous knowledge regions, increasing the risk of speculative or incoherent outputs [6]. By contrast, metaprompt-driven orchestration continuously injects relevant contextual and mnemonic anchors at runtime, constraining admissible generations and reducing reliance on dispersed latent associations.

From a governance and explainability perspective, overtuning enables a shift from opaque, parameter-centric alignment toward explicit, auditable system-level control. By encoding ethical boundaries, role constraints, and policy enforcement as external architectural components, behavioral regulation becomes inspectable and revisable without retraining [9], [10]. This separation creates new loci of explanation grounded in execution traces, constraint evaluation, and policy provenance, suggesting a complementary path for Explainable AI focused on governed behavior rather than internal model interpretability alone.

Appendix A. Illustrative Metaprompt Example

This appendix presents a minimal, non-executable example of a metaprompt structure intended solely for illustrative purposes. The objective is to demonstrate how governance constraints, role separation, and behavioral directives may be expressed as an external architectural control artifact within the overtuning paradigm. This example does not prescribe a concrete syntax nor represent an implementation specification.

```
governance:
  role: planner
  constraints:
    - no speculative claims
    - explicit uncertainty declaration
  failure_policy:
    retry_with_restricted_scope
```

This illustrative structure highlights how overtuning mechanisms can enforce behavioral boundaries and failure

handling policies without modifying internal model parameters. Such metaprompt constructs operate as transversal governance artifacts within the proposed Architecture Minimum Executable (AME), rather than as executable system components.

VIII. Acknowledgement

The author thanks Fabiana Cristina Collistet Melo for critical discussions on epistemic enclosure and circularity risks, which informed the governance analysis presented in this work.

References

[1] T. B. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, 2020.

[2] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, 2022.

[3] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, 2022.

[4] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[5] H. Chase, "Language models as agents," *arXiv preprint arXiv:2305.13245*, 2023.

[6] E. M. Bender *et al.*, "On the dangers of stochastic parrots," in *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[7] P. Liu *et al.*, "Pre-train, prompt, and predict: A systematic survey of prompting methods," *ACM Computing Surveys*, 2023.

[8] L. Reynolds and K. McDonell, "Prompt programming for large language models," *arXiv preprint arXiv:2102.07350*, 2021.

[9] M. Mitchell *et al.*, "Model cards for model reporting," in *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, 2019.

[10] S. Amershi *et al.*, "Guidelines for human-ai interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2019.