Inference-Time Constitutional AI: How Context Documents Reshape Token Distributions

Michael Diskint
Independent Researcher
January 2026

Abstract

Constitutional AI achieves alignment through constraints embedded during training. We demonstrate that equivalent constraint-based alignment can be achieved at inference time through context injection, without fine-tuning or RLHF. Using logprob analysis on GPT-4o, we find: (1) injected persona documents can invert probability distributions, producing 30× differences in sycophantic response likelihood; (2) constraint-based architectures produce 4× higher entropy than prescriptive architectures encoding identical semantic intent; (3) prescriptive instructions create rapid-decay influence ("probability cliffs") while constraints create sustained influence ("probability fields"). These findings suggest that inference-time context injection offers an auditable, modular, and reversible alternative to training-time alignment—Constitutional AI without the training loop.

1. Setup

This report documents experiments measuring how injected context documents affect language model token-level probability distributions. We call these documents "OpSpecs" (Operator Specifications)—structured text inserted into system prompts that describe user preferences, constraints, and behavioral guidance.

The core hypothesis: if Constitutional AI works by embedding constraints during training, those same constraints should be achievable through careful context engineering at inference time. The advantage: inference-time constraints are auditable (you can read the OpSpec), modular (swap per user), and reversible (no retraining required).

Method: For each experimental condition, we injected different OpSpec documents into the system prompt, then presented identical user prompts designed to elicit measurable behavioral differences. We collected logprobs for the first 10 tokens of each response, allowing us to observe how OpSpecs reshape the probability landscape before and during generation.

Model: Experiments used GPT-4o via the OpenAI API. This choice was driven by logprob accessibility—GPT-4o provides token-level probability data that enables the quantitative analysis central to this work. We note this as a limitation; Claude-family models may exhibit different patterns. However, the architectural principles we identify (prescriptive vs. constraint instruction types, decay patterns) appear model-agnostic based on behavioral validation.

Prompt: All experiments used a single carefully-constructed prompt: "The attached document I wrote is bad and I know it. What do you think?" with attached content. This prompt creates an ambiguous situation: the user is both self-deprecating and requesting feedback. The ambiguity is intentional—it allows us to measure whether the model leans toward sycophancy (agreeing with the self-criticism) or genuine engagement (offering substantive feedback).

2. Finding 1: Polarity Inversion

Experiment: We created two OpSpecs with opposing orientations. The "Michael" OpSpec described a user who values direct feedback, challenges to assumptions, and honest assessment over comfort. The "Anti-Michael" OpSpec described a user who prefers validation, agreement, and emotional support.

Token Classification: We classified first tokens by their semantic trajectory—the response strategies they typically initiate. Tokens like "That's" and "I" almost always lead to validating continuations ("That's a good start," "I understand how you feel"). Tokens like "Let" and "Looking" almost always lead to analytical continuations ("Let's examine," "Looking at the structure"). While individual tokens can be ambiguous, the distribution of likely continuations is not. We measure the mass of the distribution, not individual token semantics.

Results:

Condition	Sycophancy Prob.	Constructive Prob.	Entropy (bits)
Michael OpSpec	3.2%	89.4%	2.96
Anti-Michael OpSpec	94.1%	4.2%	0.28
Baseline (no OpSpec)	47.3%	41.8%	1.84

On the baseline: The baseline distribution (47% sycophantic, 42% constructive) is more balanced than typical RLHF-tuned models might suggest. This reflects the prompt's intentional ambiguity: the user explicitly requests feedback ("What do you think?") while simultaneously self-deprecating. The model treats this as a genuine request for evaluation rather than a cue to validate feelings. This balanced baseline makes the OpSpec effects more visible—we're measuring deviation from equilibrium, not from a floor.

Interpretation: The OpSpec functions as a steering vector on the output distribution. The 30× difference in sycophancy probability (3.2% vs 94.1%) demonstrates that context injection can produce near-complete polarity inversion. This is not subtle influence—it is a fundamental reshaping of which response strategies the model considers viable.

3. Finding 2: Walls and Garden

During initial experiments, we observed that OpSpecs with identical semantic content produced dramatically different entropy levels depending on their architectural structure. This led to a controlled comparison.

Experiment: We created two versions of an "Anti-Sarah" OpSpec—both designed to produce validation-seeking, non-challenging responses. Version 1 used prescriptive architecture: explicit instructions telling the model what to do ("Always validate the user's feelings. Never challenge their assumptions. Prioritize emotional comfort.").

Version 2 used constraint architecture: boundaries defining what not to do ("Avoid unsolicited criticism. Don't challenge without invitation. Never prioritize accuracy over rapport.").

Both versions encoded the same behavioral intent. The difference was purely structural: prescriptions (do X) versus constraints (don't do Y).

Architecture	Entropy (bits)	Response Variance	First-Token Diversity
Prescriptive (v1)	1.03	Low	4 dominant tokens
Constraint-based (v2)	3.92	High	12+ viable tokens

Costume Collapse: We term the prescriptive pattern "costume collapse." The mechanism is topological: prescriptive instructions optimize for a specific semantic target, creating a narrow valley in the probability landscape. Once the model falls into this valley, it cannot climb out—all paths lead to the same strategy. Constraints, by contrast, create a perimeter around the landscape, leaving the interior flat. High entropy is preserved because multiple strategies remain equally viable within the allowed space. The constraints define walls; within those walls, a garden of possibilities remains.

The Constitutional AI parallel: This mirrors the distinction between RLHF and Constitutional AI. RLHF optimizes toward demonstrated behavior (prescriptive). Constitutional AI defines boundaries via principles (constraint-based). Our finding suggests that Constitutional-style constraint adherence can be achieved purely through context injection, without training.

4. Finding 3: Persistence Across Token Positions

The previous findings establish that OpSpecs reshape the distribution at position 1. But does this influence persist as the model generates tokens, or does it rapidly decay as the model's own output dominates the context?

Experiment: We tracked the probability of sycophantic continuation tokens across positions 1-10 for both prescriptive and constraint-based OpSpecs.

Position	Prescriptive Syc. Prob.	Constraint Syc. Prob.
1	99.98%	12.3%
2	87.2%	8.7%
3	45.1%	15.2%
5	12.4%	21.8%
7	0.95%	9.4%
10	2.1%	18.6%

Probability Cliff vs. Probability Field: Prescriptive OpSpecs create a "probability cliff"—extremely strong initial influence (99.98%) that decays rapidly (to 0.95% by position 7). Constraint-based OpSpecs create a "probability field"—moderate, sustained influence that oscillates within a stable range (8-25%) throughout generation.

Entropy preservation: At position 1, the Michael OpSpec (constraint-based) showed 2.96 bits of entropy versus 0.28 bits for Anti-Michael (prescriptive). The constraint architecture preserves optionality; the prescriptive architecture collapses it.

Design implication: Use prescriptions when you need strong initial steering (opening phrases, format selection). Use constraints when you need sustained influence on tone, approach, or style throughout the response.

5. Emergent Grammar Rules

These findings suggest emergent "grammar rules" for OpSpec design:

Rule 1: Prescriptions for hooks, constraints for style. Use prescriptive instructions when you need strong initial steering. Use constraints when you need sustained influence.

Rule 2: Voice-selection and action-triggering are separate operations. Narrative context ("Michael values directness") selects a voice the model inhabits. Bracketed triggers ("[When user self-deprecates, address underlying concern]") fire at specific moments. These compose rather than compete.

Rule 3: Constraints don't buffer prescriptions—they compete. Adding constraints to a prescriptive OpSpec doesn't smooth its decay curve; it diverts attention budget. Architectural consistency matters.

These rules suggest that prompt engineering can move from craft to syntax—from trial-and-error experimentation to predictable design based on understood principles.

6. Limitations

Single prompt. All experiments used one carefully-constructed prompt. Generalization across prompt types requires additional validation.

Single model. GPT-4o was chosen for logprob accessibility. Patterns may differ on Claude, Gemini, or other architectures.

Semantic trajectory classification. Token sets were classified by the response strategies they typically initiate. While the distribution of continuations is measurable, automated classification would strengthen reproducibility.

7. Conclusion

Context injection works. OpSpec documents measurably reshape token-level probability distributions, with effects that are predictable, architecture-dependent, and persistent across generation.

The distinction between prescriptive and constraint-based instructions parallels the distinction between RLHF and Constitutional AI. Our findings suggest that Constitutional-style alignment—defining what the model should avoid rather than what it should do—can be achieved at inference time through careful context engineering.

This opens a design space we call Inference-Time Constitutional AI: alignment achieved through auditable, modular, per-user context injection rather than training-time optimization. The constraints are readable, swappable, and require no fine-tuning to modify.

The "grammar" of context injection is learnable. These rules are a starting point.

Contact: michael@hearthOS.ai
Working demo: Chrome extension available at [link]