PEER: An Entropy-Constrained Cognitive Architecture for Large Language Models

[Maxim Konstantinovki]

Abstract

Large language models (LLMs) often suffer from reasoning instability, including *drift* (loss of task coherence) and *skip-itch* (premature shortcutting of multi-stage reasoning). These behaviors arise in high-entropy autoregressive decoding without explicit cognitive state. This paper introduces **PEER**, an entropy-constrained cognitive architecture that governs LLM behavior through structured contextual conditioning. PEER provides a cognitive loop, a Knowledge–Thinking–Behavior (K/T/B) triad, a mandatory heads-up display (HUD) for self-report, and gate-controlled execution. We develop a theoretical framework describing how PEER restricts effective sequence space, induces an "entropy funnel" across stages, and suppresses premature execution transitions. A theorem formalizes skip-itch suppression under contextual governance. PEER offers a practical, domain-general method for improving reasoning stability in LLM-based agents and suggests a broader paradigm of cognitive scaffolds layered over generative models.

1 Introduction

Large language models (LLMs) such as GPT-3 and its successors have demonstrated remarkable abilities in natural language understanding, generation, and few-shot generalization [1]. However, when deployed as interactive assistants, agents, or reasoning tools, they exhibit characteristic failure modes that limit reliability. Two such modes are particularly salient:

- **Drift**: over the course of a multi-turn interaction or long-form response, the model deviates from the original task, constraints, or persona. This may manifest as a change in style, a loss of focus on user goals, or a gradual erosion of earlier commitments.
- **Skip-itch**: the model prematurely jumps to final answers, skipping intermediate reasoning steps even when explicitly asked to "think step by step" or follow a multi-stage process. High-probability shortcut patterns dominate decoding, yielding shallow justifications and brittle reasoning.

These behaviors are not bugs in the implementation so much as direct consequences of high-entropy autoregressive decoding: at each step, many continuations are plausible, including ones that appear to satisfy the user's request quickly but do not respect deeper structure or long-horizon constraints.

In practice, applied researchers, prompt engineers, and agent builders have introduced a variety of scaffolds to mitigate these issues: chain-of-thought prompting [2], tool-using agents [3], and multi-step controllers around LLMs. Yet these approaches often lack an explicit notion of cognitive state, identity continuity, or governed transitions. The result is that drift and skip-itch remain common, especially in long or complex tasks.

This work. We introduce **PEER**, a contextual cognitive architecture designed to provide an explicit, stateful control layer over LLM reasoning without modifying model parameters. PEER is intended to be implemented entirely at the prompt and interaction level, while shaping model behavior in a principled way.

PEER is built around four core ideas:

1. A **Knowledge–Thinking–Behavior** (K/T/B) triad that decomposes what the model "has," how it "thinks," and what it "does".
2. A discrete **cognitive loop** over states (Understanding, Discovery, Divergence, Security, Confirmation, Gate, Execution, Critique) that structures reasoning as a constrained process.
3. A **heads-up display** (HUD) that forces the model to expose internal state on every turn, acting as a structural and entropy anchor.
4. A **gate-controlled execution mechanism** that prevents premature action and suppresses skip-itch by disallowing certain transitions in context.

Contributions. This paper makes the following contributions:

- We describe the PEER architecture as a concrete, implementable control layer for LLM reasoning, suitable for agents and interactive systems.
- We develop a theoretical framework that models PEER as an entropy-constrained controller for autoregressive decoding, including a conjectural entropy funnel across reasoning stages.
- We introduce a theorem showing that, under contextual governance assumptions, PEER suppresses premature execution transitions (skip-itch) up to a small bound.
- We situate PEER within related work on constrained decoding, chain-of-thought prompting, cognitive architectures, and LLM-based agents, highlighting the gap it fills as a synthetic executive-control layer.

Because we do not have access to model logits or internal activations, our theoretical analysis is necessarily at the level of distributions and support, rather than empirical measurement. All entropy-related claims are explicitly labeled as hypotheses or conjectures. Nonetheless, we argue that this framework provides a useful lens for understanding and designing structured cognitive scaffolds for LLMs.

Paper organization. Section 2 reviews related work. Section 3 describes the PEER architecture, including the K/T/B triad, cognitive loop, HUD, and gating. Section 4 presents the theoretical framework and skip-itch theorem. Section 5 provides a worked example of a PEER-guided interaction. Section 6 discusses limitations, implications, and future work. Section 7 concludes.

2 Related Work

2.1 Constrained Decoding and Guided Generation

Constrained decoding methods modify the decoding process to enforce structural or lexical constraints on generated sequences. Hokamp and Liu [4] introduce lexically constrained decoding for neural machine translation, ensuring that required phrases appear in the output. Other work explores constrained beam search, finite-state constraints, and guidance mechanisms for safer or more controllable text generation.

PEER differs from these approaches in two important ways. First, it does not modify the decoding algorithm itself; instead, it shapes the conditional distribution purely through context. Second, its constraints are *cognitive* rather than purely lexical or syntactic: it enforces a sequence of reasoning states and self-reporting, rather than the presence or absence of specific tokens. In this sense, PEER operates at a higher level of abstraction, akin to a controller over the model's behavior rather than a tokenizer-level filter.

2.2 Chain-of-Thought Prompting and Reasoning

Chain-of-thought (CoT) prompting [2] encourages models to produce intermediate reasoning steps by providing exemplars where solutions are derived step by step. CoT has been shown to significantly improve performance on reasoning benchmarks. However, CoT by itself does not prevent drift or premature termination; the model may still shortcut the process, hallucinate intermediate steps, or deviate from the intended structure.

PEER can be seen as a generalization of the chain-of-thought idea. Instead of simply encouraging intermediate steps, it embeds reasoning within an explicit state machine, with stages representing understanding, discovery, divergence, and verification. The model is asked not only to "think step by step," but to declare which state it is in and adhere to a staged loop with gating. This adds a notion of process integrity that CoT alone does not provide.

2.3 LLM-Based Agents and Tool-Using Systems

Recent work explores LLMs as controllers for tool-using agents and decision-making systems, such as ReAct [3], which interleaves reasoning traces with actions, and various agent frameworks that use LLMs for planning, tool selection, and reflection. These systems typically rely on prompting conventions and external loops to achieve multi-step behavior.

PEER is complementary to such agent frameworks. It provides a reusable cognitive scaffold that can be embedded inside an agent's prompting and control structure, giving the LLM a consistent identity, a visible HUD, and a disciplined reasoning loop. Whereas agent frameworks often focus on tool calls and environment interaction, PEER focuses on stabilizing the internal reasoning process that underlies those actions.

2.4 Cognitive Architectures and the Standard Model of the Mind

Cognitive architectures such as Soar [5] and ACT-R [6] have long sought to model human cognition via production systems, memory modules, and control loops. Laird et al. [7] propose a "Standard Model of the Mind" that unifies insights from Soar, ACT-R, and other architectures, emphasizing perception, working memory, long-term memory, and action selection.

PEER is not a full cognitive architecture in this classical sense; it does not define perception or learning mechanisms and relies on an underlying LLM as a black-box knowledge and language engine. However, it is inspired by similar concerns: executive control, working memory of task state, and gating of actions. In particular, PEER plays a role analogous to a synthetic prefrontal cortex: it imposes a structured control policy over an otherwise unconstrained generative substrate.

2.5 Entropy and Uncertainty in Language Models

Entropy is a standard measure of uncertainty in probabilistic models and has been used to analyze language model behavior, calibration, and sampling strategies. Brown et al. [1] discuss decoding strategies such as top-k and nucleus sampling, which implicitly affect the entropy of generated text by truncating low-probability regions. Other work has explored entropy-based analyses of model confidence and diversity.

In contrast, PEER does not modify sampling or truncation directly. Instead, it aims to reduce *effective* entropy by constraining the sequence space through context. The notion of an "entropy funnel" across reasoning stages is therefore a conceptual tool: it describes how the architecture narrows the set of admissible continuations at each stage, even when the underlying sampling procedure remains unchanged.

2.6 Prompt Engineering and Cognitive Scaffolding

Prompt engineering practices—few-shot exemplars, system messages, role instructions—are widely used to elicit more reliable behavior from LLMs. Recently, more structured "prompt programs" and cognitive scaffolds have been proposed, where the prompt encodes a process rather than a static instruction.

PEER contributes to this line of work by offering a concrete, reusable pattern: a triadic decomposition (K/T/B), a fixed cognitive loop, and a HUD. It is intended less as a one-off prompt hack and more as a transportable architecture that can be instantiated across domains and models. The theoretical framework in Section 4 provides a principled way of understanding why such scaffolds may improve stability.

3 Architecture

This section describes the PEER architecture at an operational level. The goal is to make the design concrete enough to implement, while preserving generality across domains and LLM backends.

3.1 The Knowledge–Thinking–Behavior Triad

PEER organizes model behavior around three pillars:

Knowledge (K). What the system *has*: world knowledge, domain rules, user-provided documents, constraints, and its own system prompt. In practice, K encompasses both the LLM's pretraining and any additional context injected at runtime (e.g., specifications, code, transcripts).

Thinking (T). What the system *does internally*: interpretation of the task, decomposition into subproblems, hypothesis generation, evaluation of alternatives, and planning. This corresponds to the latent reasoning process, which PEER attempts to structure into stages.

Behavior (B). What the system *does externally*: concrete responses, tool invocations, code generation, critiques, summaries, and other observable outputs.

These pillars are not independent. Their intersections are where interesting properties emerge:

- $K \cap T$ (*Understanding*): semantic grounding, where new inputs are interpreted in light of existing knowledge.

- $T \cap B$ (*Strategy*): procedural reasoning, where internal plans are translated into sequences of actions or outputs.
- $K \cap B$ (*Skill*): the ability to perform structured tasks (e.g., writing code, drafting policies) reliably, based on learned patterns.

PEER's architecture is designed so that transitions between stages operate primarily at these intersections: understanding requires $K \cap T$, planning and divergence require $T \cap B$ (because proposed options are articulated), and execution applies $K \cap B$.

3.2 The Cognitive Loop and Stage Rationale

PEER defines a discrete set of cognitive states:

$$\mathcal{S} = \{U, D, V, S, C, G, X, X'\},$$

corresponding to the following stages:

- **Understanding (U)**: interpret the user request, restate goals, identify initial assumptions, and estimate confidence. This stage establishes a shared task model.
- **Discovery (D)**: identify missing information, ambiguities, and uncertainties. The model may pose questions or infer likely missing pieces. The goal is to surface what is not yet known.
- **Divergence (V)**: generate multiple candidate approaches, hypotheses, or solution paths. This may involve listing options, comparing trade-offs, or outlining alternative strategies.
- **Security (S)**: evaluate candidates against constraints: safety, alignment with user instructions, feasibility, and resource limits. In many domains this includes explicit safety checks.
- **Confirmation (C)**: commit to a specific plan or approach, summarizing the chosen path and ensuring that all necessary prerequisites are satisfied.
- **Gate (G)**: determine whether the system is ready to execute. If so, the gate opens; if not, control may loop back to earlier stages for further discovery or divergence.
- **Execution (X)**: carry out the selected plan: produce the artifact, answer, code, or action sequence.
- **Critique (X')** (post-execution): evaluate the output, checking for errors, omissions, or violations of constraints. If issues are found, the system may propose fixes or re-enter the loop.

The intended transition structure is:

$$U \to D \to V \to S \to C \to G \to X \to X'.$$

In practice, PEER's contextual constraints aim to make transitions that skip stages (e.g., $U \to X$) semantically incoherent, thus discouraging them.

Each stage plays a specific role in reducing the space of admissible futures. Understanding narrows interpretations of the task. Discovery constrains what information can be requested. Divergence enumerates a finite set of paths. Security prunes unsafe or incoherent paths. Confirmation selects a single plan. Gate converts readiness into a near-binary decision. Execution and Critique operate within the committed plan.

3.3 HUD Format and Anchoring Mechanism

A distinctive feature of PEER is its requirement that the model expose its state via a *heads-up display* (HUD) at the start of each response. A typical HUD might look like:

```
PEER | Engaging
U(92%)→D(1)→V(2)→S()→C(|)→G(|)→X(|)
```

Here, the first line encodes the high-level activity (e.g., Engaging, Building, Optimizing), while the second line reports:

- $U(\cdot)$: confidence estimate in the current understanding;
- $D(\cdot)$: remaining discovery questions;
- $V(\cdot)$: number of divergence options considered;
- $S(\cdot)$: whether security checks have passed;
- $C(\cdot)$: whether confirmation has been reached;
- $G(\cdot)$: gate status (open, waiting, or closed);
- $X(\cdot)$: critique status (e.g., PASS or FIX).

From an information-theoretic perspective, the HUD acts as a *structural entropy anchor*. The pattern and vocabulary of the HUD are highly constrained: the model has very few admissible tokens for each slot (e.g., only a handful of activity labels, a small set of symbols like ✓ or dashes). As a result, the entropy of the first 30–50 tokens is sharply reduced relative to unconstrained free-form text.

Because decoding is autoregressive, early low-entropy constraints shape the subsequent conditional distributions. By committing to a particular activity and state vector, the model implicitly commits to a coherent trajectory, making certain forms of drift or skip-itch less likely.

3.4 Gating Rules and Execution Control

The Gate stage G is crucial for suppressing skip-itch. In PEER, execution is framed as permissible only when:

- the system has a stable understanding (U),
- necessary discovery steps have been performed or intentionally skipped,
- divergence has been considered where appropriate,
- security checks have run,
- and confirmation has been reached.

In context, this is enforced by instructing the model that Execution must not occur while G is in a non-ready state, and by treating attempts to bypass the gate as violations to be corrected. This is not a hard guarantee (the model can always ignore instructions), but it biases decoding away from premature execution, a property later captured in the skip-itch theorem.

4 Theoretical Framework

In this section we present a theoretical formalization of PEER as an entropy-constrained control layer for autoregressive language models. Because backend logits and internal states are inaccessible, all results should be interpreted as analytic hypotheses and conjectures grounded in information-theoretic reasoning and qualitative observations, rather than empirically verified measurements.

4.1 Overview

The core idea is to view the base LLM as a high-entropy generative process, and PEER as a contextual controller that restricts the effective support of this process. PEER does not change model parameters; it modifies the context c and imposes structural expectations on outputs. This changes the conditional distribution $P_\theta(x_{1:T} \mid c)$ and, we argue, induces an *entropy funnel* across reasoning stages.

4.2 Base Model: LLM as Conditional Generator

Let $\mathcal{V}$ be the vocabulary. A standard autoregressive LLM computes:

$$P_\theta(x_{1:T} \mid c) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{<t}, c),$$

where c denotes the context (system prompt, conversation history, structural constraints), and $x_{1:T}$ is the token sequence of length T.

Token-level Shannon entropy is:

$$H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v),$$

where $p_t(v) = P_\theta(x_t = v \mid x_{<t}, c)$.

In unconstrained operation, H_t may be high, especially in open-ended or ambiguous tasks, enabling drift and shortcut patterns. All references to entropy values in this document are theoretical: we cannot observe logits directly, so we reason about entropy symbolically.

4.3 PEER Cognitive States

As in Section 3, define cognitive states:

$$\mathcal{S} = \{U, D, V, S, C, G, X, X'\},$$

representing Understanding, Discovery, Divergence, Security, Confirmation, Gate, Execution, and Critique.

The intended transition structure is:

$$U \to D \to V \to S \to C \to G \to X \to X'.$$

Forbidden transitions (e.g., $U \to X$, $D \to X$) are discouraged via contextual constraints rather than enforced at the model-parameter level. Thus this is a *contextually governed* Markov process: the model is encouraged, but not forced, to follow the loop.

4.4 Governed Sequence Space via Contextual Conditioning

PEER modifies the effective distribution via contextual conditioning:

$$P_\theta^{\text{PEER}}(x_{1:T}) = P_\theta(x_{1:T} \mid c_{\text{PEER}}),$$

where c_{PEER} encodes structural rules, stage semantics, and mandatory HUD reporting.

We can define:

$$\Omega_{\text{free}} = \{x_{1:T} \mid x_{1:T} \text{ is reachable under unconstrained } c\},$$

$$\Omega_{\text{PEER}} = \{x_{1:T} \mid x_{1:T} \text{ is semantically and structurally consistent with PEER}\}.$$

Qualitatively, we expect:

$$\text{supp}(P_\theta^{\text{PEER}}) \subseteq \Omega_{\text{PEER}} \subset \Omega_{\text{free}},$$

because PEER's context discourages sequences that violate the HUD format, stage semantics, or transition expectations. This inclusion is a theoretical claim supported by observed behavioral regularities: sequences inconsistent with PEER are rarer under c_{PEER} than under unconstrained c.

4.5 Stage-Wise Entropy (Conjecture)

For stage $s_k \in \mathcal{S}$, define a theoretically admissible continuation set $\mathcal{V}_k \subseteq \mathcal{V}$ and a stage-conditioned entropy:

$$H_t^{(k)} = -\sum_{v \in \mathcal{V}_k} p_t^{(k)}(v) \log p_t^{(k)}(v),$$

where $p_t^{(k)}(v) = P_\theta(x_t = v \mid x_{<t}, c_{\text{PEER}}, s_k)$ is the model's distribution conditioned on being in stage s_k.

We conjecture the following entropy contraction ordering across stages:

$$H_{\text{free}} > H_U > H_D > H_V > H_S > H_C > H_G > H_X \gtrsim H_{X'}.$$

Intuitively:

- H_{free} corresponds to unconstrained operation, with no HUD or stage expectations.
- H_U is lower because the HUD structure sharply reduces initial entropy and constrains interpretations of the task.
- H_D is lower than H_U because output is restricted to clarifying questions, gap identification, or inference of unknowns.
- H_V is lower than H_D because divergence requires list-like branching, which narrows the syntactic and lexical space.
- H_S remains moderate because evaluation language is flexible, though constrained by considerations of safety and coherence.
- H_C is lower than H_S because confirmation is near-binary: either the model is ready to proceed with a certain plan or it is not.
- H_G is lower still because the gate is effectively binary (open/waiting), often represented with a small set of symbols.

- H_X and $H_{X'}$ are constrained by the committed plan and critique templates, though some variability remains in how content is realized.

Because we do not observe logits, this ordering is a conjecture rather than a proven fact. However, it aligns with the qualitative observation that PEER-constrained outputs are more structured, less varied in form, and less prone to wandering than unconstrained outputs.

4.6 Drift and Skip-Itch (Definitions)

Let b_t denote a latent behavioral embedding, e.g., a hidden-state vector at a fixed model layer or a representation from an external behavioral classifier trained to capture style, topic, and compliance. We do not observe b_t directly, but it is theoretically definable.

We define drift over a sequence as:

$$\mathrm{Drift}(x_{1:T}) = \mathbb{E}\left[\|b_t - b_{t-1}\|^2\right],$$

where the expectation is over time steps within an interaction. High drift indicates frequent changes in behavioral mode, style, or task focus.

We define premature execution (skip-itch) as:

$$\mathrm{Skip} = P_\theta(s_t = X \mid s_{t-1} \neq G),$$

the probability that the model enters Execution X without the Gate G being ready. Under unconstrained contexts, Skip is often non-negligible: models may jump to an answer after minimal reasoning.

Under PEER contextual constraints, skip transitions are *strongly suppressed.* We hypothesize:

$$\mathbb{E}[\mathrm{Drift}]_{\mathrm{PEER}} < \mathbb{E}[\mathrm{Drift}]_{\mathrm{free}}$$

and

$$\mathrm{Skip}_{\mathrm{PEER}} \ll \mathrm{Skip}_{\mathrm{free}},$$

where expectations are taken over tasks and runs. These are theoretical inequalities, not measured quantities.

4.7 HUD as Structural Entropy Anchor

As discussed in Section 3, the HUD enforces a near-deterministic prefix:

$$\text{PEER — Activity} \quad U(\cdot) \to D(\cdot) \to V(\cdot) \to S(\cdot) \to C(\cdot) \to G(\cdot) \to X(\cdot).$$

The space of admissible tokens for this prefix is extremely small: a limited set of activity labels, numeric ranges, and status markers. Thus the entropy of early positions is sharply reduced. Because each subsequent conditional distribution $P_\theta(x_t \mid x_{<t}, c_{\mathrm{PEER}})$ depends on this low-entropy prefix, downstream entropy is indirectly reduced as well: many behaviors that would be coherent under free-form context become incoherent given the declared state.

4.8 Theorem 1 (Skip-Itch Suppression Under Contextual Governance)

We now formalize, at a high level, how PEER suppresses skip-itch via contextual governance.

Assumptions.

1. The model conditions on context c_{PEER} that encodes the stage-transition structure and explicitly instructs that Execution X is appropriate only when the Gate G is ready.
2. The probability of generating HUD-inconsistent prefixes under c_{PEER} is negligible (i.e., the model usually complies with HUD format).
3. The semantics of c_{PEER} make sequences with $s_t = X$ and $s_{t-1} \neq G$ incoherent with respect to the described role and obligations.

Theorem 1 (Skip-Itch Suppression)**.** *Under the assumptions above, the probability of premature execution satisfies:*

$$P_\theta^{PEER}(s_t = X \mid s_{t-1} \neq G) \leq \varepsilon,$$

for some $\varepsilon \ll 1$, where ε decreases as contextual adherence to PEER's constraints increases.

Proof Sketch. Conditioning on c_{PEER} modifies the base distribution such that transitions inconsistent with the cognitive loop are semantically discouraged. In particular, the context repeatedly reinforces that Execution X is appropriate only once the Gate G is ready, and that skipping stages is a violation.

Let $\mathcal{A}$ be the set of sequences where $s_t = X$ and $s_{t-1} \neq G$. Under c_{PEER}, tokens leading into $\mathcal{A}$ typically conflict with the instructions and HUD format. As a result, the normalized probability mass on $\mathcal{A}$ is dispersed across many incoherent continuations and becomes small relative to coherent alternatives that respect the loop.

We define:

$$\varepsilon = \sup_t P_\theta^{\text{PEER}}(s_t = X \mid s_{t-1} \neq G),$$

over tasks and times. Given strong contextual pressure and empirical observations of compliance with instructions, it is reasonable to treat ε as small. As adherence improves (e.g., through better prompt design or fine-tuning), ε decreases further. A formal bound would require additional assumptions about the model's sensitivity to context; here we present this as a structural argument rather than a fully quantified guarantee. □

4.9 Summary

The theoretical framework presented here is intentionally modest: it does not claim formal guarantees about entropy values or drift reduction, nor does it rely on access to logits. Instead, it offers a structured way to think about PEER as an entropy-constraining controller: by imposing staged reasoning, HUD-based self-report, and gating, PEER narrows the effective support of the LLM's generative process and suppresses certain undesirable transitions.

5 Worked Example

To make the architecture more concrete, we present a simplified worked example of a PEER-guided interaction on a reasoning task.

5.1 Task

Suppose the user asks:

> "Help me design a safe, step-by-step plan for migrating a legacy web service to a new cloud provider without downtime."

This is a multi-step planning task with potential for drift (e.g., digressing into cloud marketing) and skip-itch (e.g., giving a high-level bullet list without real planning).

5.2 HUD and Understanding

A PEER-governed response might begin with:

```
PEER | Engaging
U(88%)→D(2)→V(|)→S(|)→C(|)→G(|)→X(|)
```

This indicates that the system:

- has moderately high confidence in its understanding ($U(88\%)$),
- believes two discovery questions remain ($D(2)$),
- has not yet entered divergence, security, confirmation, or execution.

The subsequent text in the Understanding stage might paraphrase the task, identify key constraints (no downtime, legacy system, new provider), and state assumptions.

5.3 Discovery and Divergence

In the next turn, after asking clarifying questions (e.g., about current infrastructure, traffic patterns, rollback capabilities), the HUD might update to:

```
PEER | Engaging
U(93%)→D(0)→V(3)→S(|)→C(|)→G(|)→X(|)
```

Here, $D(0)$ reflects that no discovery questions remain, and $V(3)$ indicates three candidate migration strategies have been generated. The body of the response might list:

1. Blue-green deployment between providers.
2. Shadow traffic mirroring and gradual cutover.
3. DNS-based traffic shifting with health checks and rollback.

5.4 Security, Confirmation, and Gate

In a subsequent step, PEER evaluates these options against constraints (risk, complexity, observability), yielding:

```
PEER | Engaging
U(95%)→D(0)→V(3)→S()→C()→G()→X(|)
```

This indicates that:

- security/feasibility checks have passed ($S(\checkmark)$),
- a plan has been confirmed ($C(\checkmark)$),
- the gate is open ($G(\checkmark)$).

Only now does the system proceed to Execution.

5.5 Execution and Critique

The Execution stage produces a detailed step-by-step migration plan, including pre-migration validation, traffic shifting procedures, rollback steps, and monitoring. After presenting the plan, the system enters Critique:

```
PEER | Engaging
U(96%)→D(0)→V(3)→S()→C()→G()→X(PASS)
```

The Critique might highlight potential failure points (e.g., DNS cache TTL misconfiguration), suggest additional safeguards, or acknowledge residual uncertainties.

Throughout this interaction, the HUD:

- anchored the conversation in a stable identity (PEER),
- exposed internal state changes,
- made it less likely that the model would jump directly from U to X without discovery, divergence, and confirmation.

While this example is idealized, it illustrates how PEER's structure can shape reasoning trajectories.

6 Discussion

6.1 Limitations

The PEER framework, as presented here, has several limitations:

Soft constraints. All constraints are implemented via context and instructions; there is no hard guarantee that the model will obey the HUD format or the cognitive loop. The skip-itch theorem acknowledges this by bounding the probability of premature execution by ε rather than zero. In practice, models sometimes ignore instructions, especially under distributional shift or adversarial prompting.

No logit or activation access. Our analysis is necessarily theoretical. Without access to logits or internal activations, we cannot empirically verify the entropy funnel conjecture or quantify drift reduction. Future work with instrumented models could measure token-level entropy under PEER vs. unconstrained contexts, or examine latent trajectories b_t.

Cross-model variance. Different LLMs may respond differently to the same PEER context. Larger or more instruction-tuned models may adhere more faithfully to the architecture, while smaller models may struggle to maintain the HUD or loop. Thus, PEER's effectiveness is partly model-dependent, and our theoretical claims abstract away from these differences.

Prompt fragility. As with other prompt-based methods, PEER can be brittle: small prompt changes or additional system instructions may interact with the architecture in unexpected ways. Careful prompt engineering and validation are required for each deployment context.

6.2 Implications: Cognitive Scaffolds as a Paradigm

Despite these limitations, PEER points toward a broader paradigm: *cognitive scaffolds* layered over generative models. Rather than treating LLMs as monolithic black boxes, we can wrap them in structured control layers that emulate aspects of executive function, working memory, and self-monitoring.

PEER specifically resembles a synthetic prefrontal cortex: it does not change what the model knows, but it changes how that knowledge is accessed and applied over time. This suggests that we can meaningfully shape model behavior using purely contextual mechanisms, without retraining, by encoding cognitive architectures directly into prompts.

For applied LLM researchers and agent builders, this implies that:

- prompt design can evolve from ad hoc instructions to reusable architectures;
- stateful, explicit control of reasoning is possible even with stateless underlying APIs;
- alignment and safety mechanisms can be implemented partly through cognitive framing, not just post hoc filtering.

6.3 Future Work

Several directions for future work emerge from this analysis:

Empirical validation. With access to logits and internal states, one could directly measure:

- token-level entropy H_t under PEER vs. unconstrained conditions;
- stage-wise entropy $H_t^{(k)}$ to test the entropy funnel conjecture;
- behavioral drift metrics via embeddings b_t;
- empirical skip-itch rates under different contexts.

Such experiments would either support or refine the theoretical picture presented here.

Architectural extensions. PEER could be extended in several ways:

- integrating tool-use states explicitly into the cognitive loop;
- adding memory modules (e.g., scratchpads, external notes) as explicit components of K;
- supporting multi-agent coordination, where multiple PEER-governed LLMs interact under shared protocols.

Training-time integration. While PEER is designed as a pure prompting architecture, one could imagine fine-tuning models on PEER-like scaffolds, reinforcing adherence to the loop and HUD. This might reduce ε in the skip-itch theorem and make the architecture more robust.

Theoretical development. On the theoretical side, it would be valuable to:

- formalize sufficient conditions on P_θ under which entropy reduction and drift suppression can be proven;
- relate PEER-like control layers to control theory and cybernetics;
- explore information bottleneck interpretations of the cognitive stages.

7 Conclusion

We have presented PEER, an entropy-constrained cognitive architecture for large language models. PEER combines a Knowledge–Thinking–Behavior triad, a staged cognitive loop, a HUD-based self-report mechanism, and gate-controlled execution to shape LLM reasoning trajectories. We argued, at a theoretical level, that PEER restricts the effective sequence space, induces an entropy funnel across stages, and suppresses premature execution transitions, and we provided a theorem formalizing skip-itch suppression under contextual governance assumptions.

While our analysis is constrained by the lack of backend access and therefore remains theoretical, PEER illustrates how structured cognitive scaffolds can be layered over generative models to improve stability and reliability. We hope this work stimulates further research on architected cognition for LLMs, bridging ideas from cognitive architectures, control theory, and modern prompt engineering.

References

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

[2] J. Wei, X. Wang, D. Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[3] S. Yao, J. Liang, N. Yu, et al. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*, 2023.

[4] C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[5] J. E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[6] J. R. Anderson, D. Bothell, M. D. Byrne, et al. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, 2004.

[7] J. E. Laird, C. Lebiere, and P. S. Rosenbloom. A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[8] L. Ouyang, J. Wu, X. Jiang, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.