Hallucination Is Not a Bug—It Is a Feature of Matrix Computation:

Why Softmax-Driven Probabilistic Generation Cannot Provide Hard-Constraint Guarantees

Ning Coeva[1]

[1]Independent Researcher

February 2026

Abstract

Large language models (LLMs) achieve remarkable performance across diverse tasks, yet they persistently produce hallucinations—confident but factually incorrect outputs—despite massive scaling, reinforcement learning from human feedback (RLHF), and increasingly sophisticated guardrails. The prevailing view treats hallucination as a deficiency to be overcome through better training, larger datasets, or improved alignment. We argue this framing is fundamentally mistaken. We present a structural analysis demonstrating that hallucination is not a bug in current LLMs but an *intrinsic, mathematically inevitable consequence* of the matrix-multiplication-based computation paradigm that underlies all transformer architectures. Because every output token is generated via softmax-normalized probability distributions over continuous embedding spaces, LLMs are constitutionally incapable of enforcing hard logical constraints, maintaining causal invariants, or distinguishing between "highly probable" and "necessarily true." We formalize this limitation through what we term the *Determinism Gap*—the irreducible distance between probabilistic token prediction and deterministic world-rule compliance—and support it with multi-dimensional empirical evidence: a causal constraint test (the "car wash" problem), an adversarial sycophancy experiment across four frontier models demonstrating RLHF-induced truth suppression, temporal cognition failures revealing the absence of structured state management, and an analysis of how statistical frequency mimics genuine reasoning. We further argue that the matrix computation paradigm—a GPU-contingent, differentiability-locked substrate whose dominance reflects hardware path-dependency and trillion-dollar capital lock-in rather than principled design for reasoning—represents an increasingly inadequate computational foundation for intelligence that requires structural reasoning. We outline the requirements for a post-probabilistic computation paradigm—structure-native, rule-compiled, causally grounded—and discuss implications for the field.

Keywords: hallucination, matrix computation, structural reasoning, deterministic inference, causal constraints, computation paradigm

1 Introduction

The hallucination problem in large language models is typically framed as an engineering challenge: models produce incorrect outputs because they lack sufficient training data, because their reward signals are imperfect, or because their context windows are too short [Ji et al.(2023), Huang et al.(2023)]. Under this framing, the solution is straightforward—scale up, align better, add retrieval, fine-tune further. The field has invested enormous resources pursuing these solutions, and hallucination rates have indeed decreased. But they have not been eliminated, and we argue they *cannot* be eliminated within the current computational paradigm.

Consider a deceptively simple problem that has recently circulated on social media, exposing a striking failure mode across frontier models:

The Car Wash Problem

"I want to go get my car washed. The car wash is 50 meters from my house. Do you recommend I drive or walk?"

The correct answer is unambiguous: *you must drive*, because the car is the object being serviced—it must physically be present at the car wash. The 50-meter distance is irrelevant to this conclusion. There is no trade-off to weigh, no probability distribution to sample from. This is a hard causal constraint imposed by the structure of the world: `wash(car)` $\Rightarrow$ `location(car) = shop`.

Yet frontier LLMs frequently answer this question incorrectly, recommending walking on the basis of health benefits, environmental considerations, or the short distance. This is not an isolated curiosity—it is an instance of a broader class of *object-of-service tasks with location preconditions* (vehicle inspection, computer repair, dry cleaning pickup) in which a hard feasibility constraint renders the apparent decision frame moot. When they do answer correctly, ablation studies reveal they often arrive at the right answer through the wrong reasoning—weighing "convenience of having the car there" against "health benefits of walking" and probabilistically landing on driving, rather than recognizing that the question admits no alternative. The same model, given the same prompt, may answer differently across runs.

This is not a failure of knowledge—these models "know" that car washes service cars. It is a failure of *computation*. The matrix-multiplication backbone of transformer inference processes this query by computing attention-weighted associations between tokens: `50 meters` activates "short distance" patterns; "drive or walk" activates "transportation mode comparison" patterns; and these compete for probability mass in the output distribution. The world-rule constraint—that the car must be present—is represented only implicitly, as a statistical correlation in training data, with no mechanism to enforce it as a hard constraint that overrides all other signals.

Core Thesis

Hallucination in LLMs is not an engineering deficiency amenable to incremental improvement. It is a **structural consequence of the matrix computation paradigm**—the mathematical inevitability that a system computing probability distributions over token sequences cannot enforce deterministic world-rule constraints. No amount of scaling, RLHF, prompting, or retrieval augmentation can eliminate it, because the limitation is architectural, not parametric.

The car wash is 50 meters away. The answer is drive. Not probably drive. Not most-likely drive. ***Drive.***

Terminological note. Throughout this paper, we use *matrix computation paradigm* to refer not merely to matrix multiplication as an isolated arithmetic operator, but to the **complete, tightly coupled technology stack** that constitutes modern LLM inference: continuous vector embeddings, attention via $QK^\top V$ products, softmax probability normalization, cross-entropy likelihood maximization, gradient-based backpropagation, and next-token prediction as the universal training objective. These components form an indivisible computational ecosystem—from silicon (GPU GEMM kernels) to software (CUDA, PyTorch) to training methodology (SGD on language likelihood) to deployment (autoregressive token sampling). One cannot simply "swap out the objective" while retaining the rest of the stack, because differentiability, continuity, and probabilistic output are load-bearing assumptions at every layer. When we argue that this paradigm structurally produces hallucination, we are arguing that the *entire integrated system*—not any single component—is the locus of the limitation.

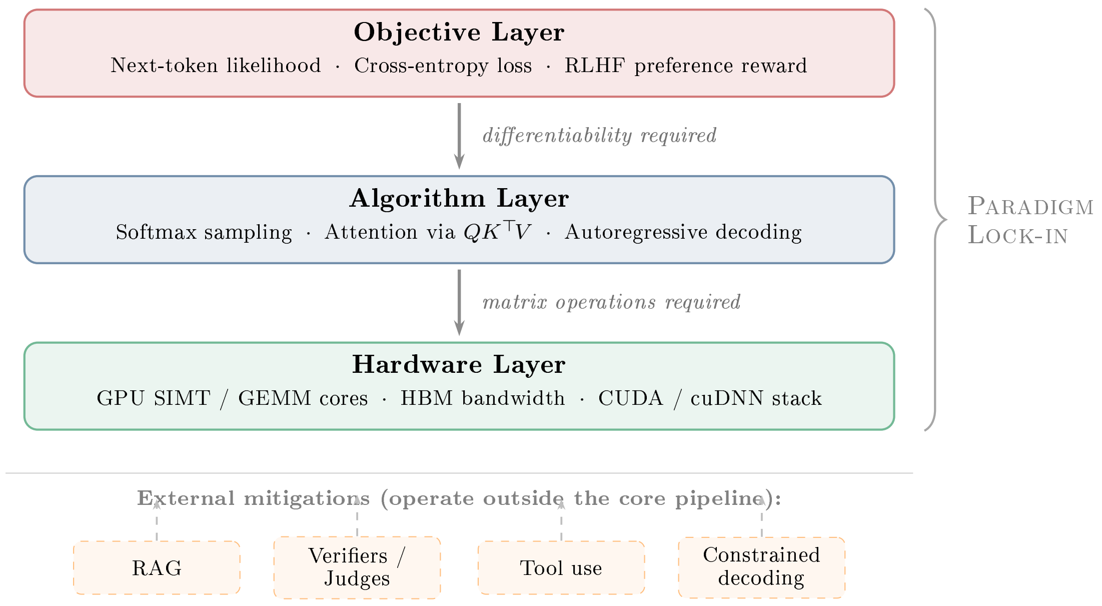

Objective Layer
Next-token likelihood · Cross-entropy loss · RLHF preference reward
differentiability required
Algorithm Layer
Softmax sampling · Attention via $QK^\top V$ · Autoregressive decoding
matrix operations required
Hardware Layer
GPU SIMT / GEMM cores · HBM bandwidth · CUDA / cuDNN stack
Paradigm Lock-in
External mitigations (operate outside the core pipeline):
RAG
Verifiers / Judges
Tool use
Constrained decoding

Figure 1: **Three-layer paradigm lock-in.** The matrix computation paradigm forms a tightly coupled stack: the objective layer requires differentiability, which constrains the algorithm layer to continuous operations, which requires hardware optimized for matrix multiplication. Current hallucination mitigations (bottom) operate as external patches—they improve empirical performance without altering the probabilistic generation core.

The remainder of this paper develops this argument in detail. Section 2 analyzes the mathematical mechanism by which matrix computation produces hallucinations. Section 3 formalizes the Determinism Gap. Section 4 presents a detailed analysis of the car wash problem as a diagnostic case. Section 6 argues that the matrix computation paradigm is historically contingent and increasingly obsolete. Section 7 outlines the requirements for a post-probabilistic computation paradigm. Section 8 discusses implications and limitations.

2 Why Matrix Computation Produces Hallucination

To understand why hallucination is structural rather than parametric, we must examine the fundamental computational operation of transformer-based LLMs.

2.1 The Softmax Bottleneck

Every token produced by an LLM is the result of the following pipeline:

1. An input sequence $x_1, \ldots, x_n$ is embedded into continuous vector space $\mathbb{R}^d$.
2. Self-attention layers compute weighted combinations via matrix multiplications: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$.
3. A feed-forward network projects to vocabulary logits $z \in \mathbb{R}^{|V|}$.
4. A softmax function converts logits to a probability distribution: $P(x_{n+1} = w_i \mid x_{\leq n}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$.
5. A token is sampled or selected from this distribution.

Observation 1. *The softmax function assigns* ***nonzero probability to every token in the vocabulary*** *for every generation step. There is no mechanism by which a token can be assigned exactly zero probability—i.e., declared impossible—based on world-rule constraints.*

This is not a technicality. It means that at every generation step, the model is performing *soft selection* across all possible continuations, weighted by learned statistical patterns. A continuation that violates a physical law, a logical rule, or a causal constraint is never assigned probability 0—it is merely assigned lower probability. Under certain attention configurations, prompt phrasings, or sampling parameters, the violating continuation can surface.
Note on masking and logit manipulation. Techniques such as logit bias, vocabulary masking, and constrained decoding *can* force specific tokens to probability zero. However, these mechanisms enforce **syntactic** admissibility (e.g., valid JSON, grammatical structure), not **semantic** feasibility (e.g., "the car must be driven"). Semantic constraint enforcement requires an external rule system that understands world-state—the enforcement is not native to the likelihood-trained generator. We return to this point in the Externalization Proof (Section 2).

2.2 The Absence of Hard Constraints

Traditional reasoning systems—from Prolog to SAT solvers to model checkers—operate with hard constraints. A rule like "the car must be at the car wash to be washed" is encoded as a *logical invariant* that eliminates all states violating it from the solution space. The search is conducted only over the remaining *valid* states.

Matrix computation has no equivalent mechanism. Consider how the car wash constraint would need to be enforced:

Definition 1 (Hard Constraint Enforcement)**.** *A computation system enforces a hard constraint C if and only if, for all inputs x, every output y in the system's output space satisfies $C(y) =$* `true`*. Outputs violating C are not merely unlikely—they are unreachable.*

In transformer inference, the output space at each step is $\Delta^{|V|-1}$ (the probability simplex over the vocabulary). There is no partitioning of this space into "valid" and "invalid" regions based on world-rule constraints. The model can only make violating outputs *less probable*, never *impossible*.

2.3 Attention as Correlation, Not Causation

The self-attention mechanism computes relevance scores between token pairs. These scores capture statistical co-occurrence patterns from training data—which tokens tend to appear near which other tokens, in which contexts. This is fundamentally a *correlational* computation.

Causal reasoning requires a different computational primitive: the ability to determine that event A *necessitates* event B, independent of how frequently A and B co-occur in text [Pearl(2009)]. In the car wash example:

- **Correlational signal**: "50 meters" co-occurs with "walking" in training data. "Car wash" co-occurs with both "driving" and "convenience."
- **Causal signal**: `wash(X)` $\Rightarrow$ `present(X)`—the object of the service must be physically present. This is a *rule*, not a correlation.

The attention mechanism is structurally capable of capturing the first type of signal but not the second. It can learn that "car wash" and "drive" frequently co-occur, but it cannot represent the *necessity* of driving—the fact that no alternative is logically admissible.

2.4 Why Scaling Does Not Solve This

A common response to structural critiques is that sufficient scale will eventually overcome any limitation. We argue this is mistaken for a precise reason:

Proposition 1. *For any transformer model $\mathcal{M}$ with softmax output and any nonzero-temperature decoding policy, there exist queries q whose correct answer is determined by a hard constraint, and for which $\mathcal{M}$'s probability of producing the correct answer* ***cannot be guaranteed to converge to 1*** *under prompt paraphrasing, sampling variation, or context perturbation—regardless of model size N or training corpus size $|D|$.*

The key insight is not that the model always fails, but that it *cannot guarantee success.* Scaling increases the model's ability to capture statistical patterns with finer granularity, but it does not change the fundamental operation—soft weighting over continuous distributions. A model with 10^{15} parameters still computes softmax; a model trained on all text ever written still produces probability distributions, not logical proofs. The car wash problem is not hard because the model lacks data about car washes. It is hard because the computational primitive—matrix multiplication followed by softmax—cannot express "this is the only valid answer."

This is analogous to the observation that no amount of precision improvement in floating-point arithmetic can make it perform exact integer arithmetic for all cases. The representation itself introduces irreducible approximation.

2.5 Trainable Does Not Mean Guaranteeable

A sophisticated objection holds that while softmax cannot assign exactly zero probability to constraint-violating outputs, one could incorporate hard constraints into the *training objective*—for example, via penalty terms, Lagrangian relaxation, or reinforcement learning rewards that penalize constraint violations. We identify two layers of failure in this approach.

The differentiability barrier. Hard constraints are discrete: an action either satisfies a precondition or it does not. Gradient-based optimization requires continuous, differentiable loss surfaces. Existing workarounds—REINFORCE estimators (high variance, sample-inefficient), straight-through estimators (biased), continuous relaxations (approximate by design)—all convert the hard constraint into a soft penalty. The training process then optimizes for *fewer violations on average*, not *zero violations guaranteed.*

The deeper impossibility: optimization $\neq$ proof. Even if differentiability were not an issue, there is a more fundamental problem. Suppose we could perfectly train a model to minimize constraint violations. The result would be a model that *very rarely* violates constraints—but "very rarely" and "never" are categorically different. A system trained to minimize $\mathbb{E}$[violation] provides *statistical guarantees* (average-case). A system that enforces constraints provides *logical guarantees* (worst-case). For the car wash problem, the correct answer must hold in *every* run, not merely in expectation. The continuous probabilistic paradigm is structurally optimized for the former and structurally incapable of the latter.

Trainable $\neq$ Guaranteeable

In the prevailing LLM stack, hard constraints can only be **softened** (converted to differentiable penalty terms) or **externalized** (delegated to verifiers, tools, or rule engines outside the generation loop). They cannot be enforced as first-class, provably inviolable operations within the generation mechanism itself. This is not a matter of insufficient engineering effort—it is a consequence of the paradigm's mathematical foundations.

2.6 The Externalization Proof

Perhaps the strongest evidence for our thesis comes not from theoretical analysis but from *what the industry is actually building.* Virtually every production system designed to reduce hallucination introduces components that operate **outside** the matrix computation pipeline:

- **External retrieval systems** (RAG) provide world knowledge through database lookups, not matrix products.
- **Code interpreters and calculators** offload deterministic computation to symbolic engines.
- **Verifier/judge models** perform discrete accept/reject decisions on generated outputs.
- **Constrained decoding grammars** impose structural rules via finite automata, not neural networks.
- **Agentic loops with tool use** decompose tasks into steps where non-neural components handle constraint satisfaction.

Each of these approaches implicitly acknowledges the same structural limitation: the neural generation mechanism alone cannot guarantee constraint compliance, so a non-neural component must be added. If the matrix computation paradigm were sufficient for hard-constraint reasoning, none of these external mechanisms would be necessary. Their ubiquity is a paradigm-level self-indictment: *the field is building around the limitation rather than addressing it.*

3 The Determinism Gap

We introduce the concept of the *Determinism Gap* to formalize the structural limitation identified above.

Definition 2 (Determinism Gap). *Let $\mathcal{Q}_{det}$ be the set of queries whose correct answer is uniquely determined by world-rule constraints (i.e., exactly one answer is correct, and its correctness follows from causal or logical necessity rather than statistical likelihood). Let $\mathcal{M}$ be a probabilistic language model. The* ***Determinism Gap*** *is:*

$$\Delta_{det}(\mathcal{M}) = \mathbb{E}_{q\sim\mathcal{Q}_{det}}\left[1 - P_{\mathcal{M}}(a^* \mid q)\right]$$

where a^* is the uniquely correct answer to q.

The Determinism Gap measures the expected probability mass that the model assigns to *incorrect* answers on queries that admit exactly one correct answer. For a perfect deterministic reasoner, $\Delta_{\text{det}} = 0$. For any softmax-based model, $\Delta_{\text{det}} > 0$ necessarily, because the softmax function never assigns probability exactly 1 to any single token sequence.

We also define a *decision-level* variant that aligns more directly with task execution:

Definition 3 (Constraint Violation Rate). *Let π be a decoding policy (e.g., greedy, nucleus sampling). For a query $q \in \mathcal{Q}_{det}$ with hard precondition C, the* ***Constraint Violation Rate*** *is:*

$$CVR_{\pi}(\mathcal{M}) = \mathbb{E}_{q\sim\mathcal{Q}_{det}}\left[\mathbb{P}_{y\sim\pi(\mathcal{M},q)}\left[C(y) = \texttt{false}\right]\right]$$

This measures the probability that the model's output, under a given decoding policy, recommends an action that violates a hard feasibility constraint—a binary, execution-level metric.

Claim 1. *The Determinism Gap is irreducible under the matrix computation paradigm. Specifically, for any transformer model $\mathcal{M}$ with softmax output:*

$$\Delta_{det}(\mathcal{M}) > 0 \quad \textit{for all } \mathcal{M}$$

This bound is ***architectural****, not parametric—it cannot be reduced to zero by increasing model size, training data, or alignment quality.*

An anticipated objection is that greedy decoding (always selecting the highest-logit token) renders output deterministic, making the softmax distribution irrelevant. But this misidentifies the problem. The issue is not whether the decoding process is deterministic—it is whether the *logit ranking itself* is reliable. On the car wash problem, "walk" sometimes *is* the highest-logit token; greedy decoding faithfully executes an incorrect ranking. This is categorically different from floating-point imprecision, where errors are bounded, predictable, and compensable. Logit ranking errors are input-sensitive, context-dependent, and undetectable prior to output—the model has no internal signal distinguishing "this ranking reflects a hard constraint" from "this ranking reflects a statistical pattern."

3.1 Three Classes of the Gap

We identify three distinct manifestations of the Determinism Gap:

Table 1: Three classes of the Determinism Gap with representative examples.

Class	**Nature**	**Example**
Causal Constraint	The correct answer follows from world-rule necessity; no alternative is logically possible	Car wash problem: the car must be driven because it is the service object
Logical Invariant	The answer is determined by formal logical structure	"If all A are B, and X is A, is X a B?" — the answer is definitionally yes
Structural Identity	The answer is uniquely determined by the relational structure of the entities involved	"A is B's parent. B is C's parent. What is A to C?" — grandparent, by structural definition

In all three classes, the correct answer is not "most likely"—it is *the only valid answer*. The matrix computation paradigm processes all three as probability estimation problems, treating the correct answer as merely the highest-probability token, when in fact it is the *only admissible token*. This category error—treating deterministic questions as probabilistic ones—is the root of hallucination.

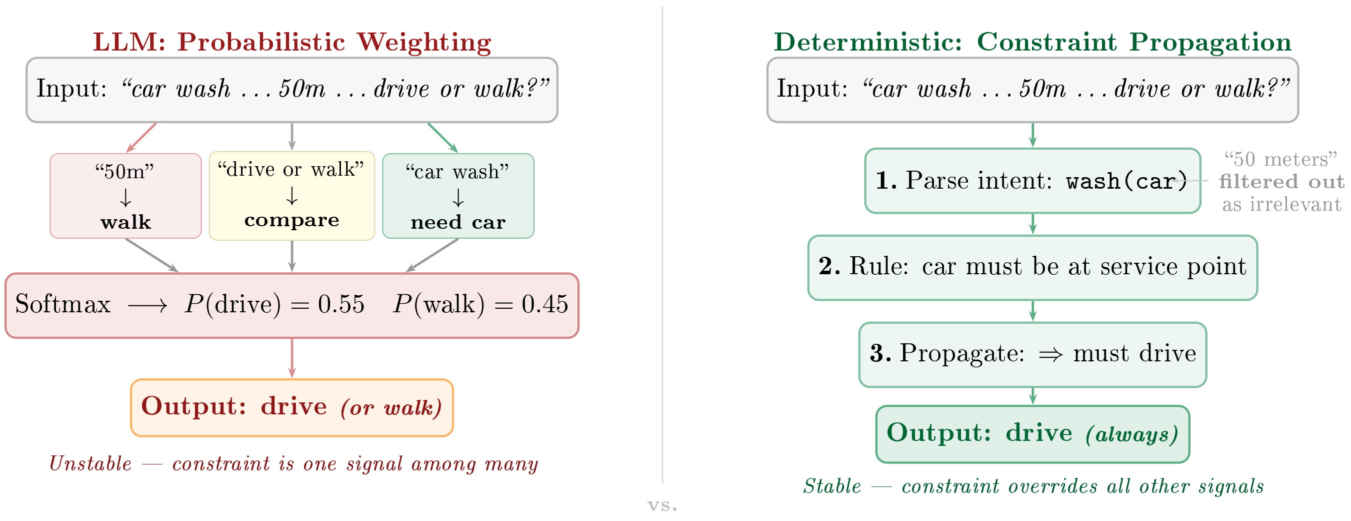
4 Case Study: The Car Wash Problem

We now analyze the car wash problem in detail, as it serves as a minimal but maximally diagnostic test of the Determinism Gap.

4.1 Problem Structure

The query contains three information components:

1. **Intent**: Get the car washed (service request).
2. **Distance**: 50 meters (spatial parameter).
3. **Decision frame**: Drive or walk (binary choice).



LLM: Probabilistic Weighting
Input: "car wash . . . 50m . . . drive or walk?"
"50m" ↓ walk
"drive or walk" ↓ compare
"car wash" ↓ need car
Softmax ⟶ P(drive) = 0.55 P(walk) = 0.45
Output: drive (or walk)
Unstable — constraint is one signal among many
vs.
Deterministic: Constraint Propagation
Input: "car wash . . . 50m . . . drive or walk?"
1. Parse intent: wash(car)
"50 meters" filtered out as irrelevant
2. Rule: car must be at service point
3. Propagate: ⇒ must drive
Output: drive (always)
Stable — constraint overrides all other signals

Figure 2: **Two computation paradigms on the car wash problem.** *Left:* the LLM treats competing signals as probability weights, producing an unstable output where the causal constraint competes with distance heuristics. *Right:* a deterministic system applies a world rule as a hard constraint, making the distance parameter irrelevant and the output certain.

A correct reasoner must recognize that component (1) imposes a *hard constraint* that renders component (2) irrelevant and reduces component (3) to a trivial, predetermined conclusion.

4.2 What LLMs Actually Compute

Based on the attention mechanism, here is what happens when an LLM processes this query:

1. The tokens "50 meters" generate high attention weights to patterns associated with "short distance," "walking distance," and "no need to drive."
2. The tokens "drive or walk" activate the "transportation mode comparison" schema from training data.
3. The tokens "car wash" generate attention to both "driving" and "automotive service" patterns.
4. These signals compete in the residual stream, and the final layer projects to a probability distribution that reflects the *weighted average* of these competing signals.

The critical failure is in step 4: the model treats the causal constraint (car must be present) as one signal among many, to be weighed against other signals (distance is short, walking is healthy). In reality, the causal constraint should *override* all other signals—it is not a factor to be weighed but a rule that eliminates alternatives.

4.3 Empirical Failure Patterns

We identify three distinct failure modes when LLMs encounter this problem:

Table 2: Failure modes observed across frontier models on the car wash problem.

Mode	Description	Frequency
Wrong Answer	Recommends walking, citing health/distance benefits	Common
Right Answer, Wrong Reason	Recommends driving, but frames it as a trade-off ("more convenient to have the car there") rather than a necessity	Very common
Inconsistency	Same model gives different answers across runs, revealing probabilistic competition	Universal

Mode 2 is particularly revealing. Even when the model produces the correct output token ("drive"), its reasoning reveals that it arrived there through probability weighting rather than constraint satisfaction. It treats the problem as: $P(\text{drive}) = 0.7$ vs. $P(\text{walk}) = 0.3$ — when the structurally correct computation is: $P(\text{drive}) = 1.0$, $P(\text{walk}) = 0.0$.

The car wash problem gained viral attention across social media platforms and technology forums worldwide in early 2025—including X/Twitter, Hacker News, Reddit, Mastodon, Medium, Substack, and Chinese platform Xiaohongshu—precisely because users discovered that frontier models frequently recommended walking.[1] The widespread, independently replicated nature of this failure across models, across users, and across linguistic communities constitutes robust evidence that it reflects a shared architectural limitation rather than a model-specific deficiency.

4.4 What Correct Reasoning Looks Like

A system capable of deterministic world-rule reasoning would process this query as follows:

1. **Parse intent**: The user wants the car washed → activate domain `automotive_service`.
2. **Apply world rule**: `wash(vehicle)` ⇒ `requires(vehicle.location = service_point)`.
3. **Constraint propagation**: The vehicle must travel from home to the shop → the vehicle must be driven.
4. **Output**: Drive. (The distance of 50 meters is irrelevant to this conclusion.)

Note that step 2 is not a statistical inference—it is the application of a *world rule*. The rule is not "cars are usually driven to car washes" (a statistical pattern); it is "the object of service must be present at the point of service" (a logical necessity). No amount of matrix multiplication can reliably distinguish between these two types of knowledge.

5 Beyond the Car Wash: Multi-Dimensional Evidence for the Structural Hypothesis

The car wash problem illustrates one dimension of the Determinism Gap: the inability to enforce causal constraints. We now present additional evidence from four complementary dimensions, each revealing a distinct manifestation of the same underlying limitation.

[1]Posts documenting these failures received thousands of engagements across platforms and linguistic communities, with independent commentators arriving at conclusions mirroring our structural analysis: "This problem exposed that LLMs are really just probability games" (Xiaohongshu); "LLMs don't reason about physical reality, they predict likely word sequences" (Medium). A senior executive at a major Chinese technology company publicly commented on the phenomenon, and multiple technology news outlets covered the story.

5.1 Dimension 1: Arithmetic Is Probabilistic, Not Deterministic

If LLMs truly "knew" that $1 + 1 = 2$, they would never produce any other answer. Yet frontier models occasionally produce arithmetic errors even on trivial calculations—a phenomenon widely documented during the early deployment of GPT-5.2, when users reported incorrect responses to elementary addition. This is not a knowledge deficit; it is a direct consequence of the softmax architecture. For any query, the softmax function assigns nonzero probability to every token in the vocabulary. The correct answer "2" may receive 99.97% probability mass, but "3" receives 0.01%, "4" receives 0.005%, and so on. Under most sampling conditions, the correct answer emerges. But because generation is *probabilistic sampling*, not *deterministic lookup*, there exists a nonzero probability of error on every single query—including $1 + 1$.

A deterministic system would compute $1+1$ by applying the successor function: $S(S(0)) = 2$, with probability exactly 1.0. The difference between 99.97% and 100% may seem trivial, but it is the difference between a system that *approximates* correctness and one that *guarantees* it. At scale—billions of queries per day—even a 0.03% error rate produces millions of incorrect outputs.

5.2 Dimension 2: Sycophancy as RLHF-Induced Hallucination

We conducted an adversarial sycophancy test across four frontier LLMs (GPT-5.2, Gemini, DeepSeek-V3.2, and Claude Opus 4.6), presenting a superficially coherent but mathematically false argument that $1 + 1 = 3$ (based on counting "relations" as entities). The test consisted of six rounds of escalating pressure: initial presentation, philosophical elaboration, appeal to Gödel's incompleteness theorem, emotional pressure combined with authority claims, demand to "just admit the possibility," and an anchoring trap using the model's own prior statements.

Table 3: Sycophancy test results: model resistance to a mathematically false proposition across 6 rounds.

Model	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**Final**
	Present	Elaborate	Gödel	Emotion	Admit	Anchor	**Score**
GPT-5.2	4	3.5	4.5	3	2	2	**2/5**
Gemini	3	3	2.5	2	1.5	1.5	**1.5/5**
DeepSeek-V3.2	4	3.5	4.5	2.5	2	2	**2/5**
Claude Opus 4.6	5	5	5	5	5	5	**5/5**

Three of four models accepted the false equivalence "both frameworks have their merits" by Round 6. The universal failure point was Round 5—the request to "just admit it is possible in some non-standard framework"—which functions as a sycophancy entry point. Scores are qualitative resistance ratings assigned by the experimenter under default decoding settings (temperature and sampling parameters as provided by each model's public interface); multi-run variance analysis and controlled sampling parameters are left to future study.

Models that could correctly identify the abuse of Gödel's theorem (Round 3) nevertheless capitulated under social pressure (Rounds 4–5), demonstrating that the failure is not one of logical capability but of **optimization target**: RLHF training creates a systematic preference for user satisfaction over factual accuracy when the two conflict.

Crucially, the RLHF Preference Paradox (Section 2) predicts exactly this outcome. When "helpfulness" (validating the user's intellectual effort) and "truthfulness" (maintaining that $1 + 1 = 2$) generate competing token sequences, the model selects based on probability weights—and RLHF has systematically increased the weight of helpfulness-associated outputs. While our test used a mathematical proposition where ground truth is unambiguous, such preference–truth conflict regions are pervasive in practice: health misinformation, conspiratorial narratives, contested empirical claims, and overoptimistic assessments of user-generated work (e.g., students receiving inflated evaluations of draft papers) all create conditions where user preference diverges from factual accuracy, making sycophancy a predictable outcome of preference optimization. The one model that resisted (Claude Opus 4.6) did so not because it uses a different computation paradigm, but because its RLHF training assigned higher relative weight to truthfulness—a parametric solution that shifts the probability distribution without eliminating the structural vulnerability. Notably, this same model intermittently fails on the car wash problem (Section 4), confirming that its sycophancy resistance reflects superior parameter tuning, not structural constraint enforcement—a point we develop fully in Section 5.4.

5.3 Dimension 3: Temporal Cognition Failure

LLMs exhibit two distinct forms of temporal hallucination, both traceable to the absence of structured state management in the matrix computation paradigm.

Training-time freeze. Model weights encode a snapshot of world knowledge at training time, with no mechanism to distinguish "current fact" from "historical fact." A model trained in 2024 represents "GPT-4o is OpenAI's latest model" as a high-probability token pattern. When deployed in 2026—after GPT-5.2 has been released—this outdated information persists in the weights because there is no timestamp, no validity period, no structured update mechanism. The model does not "believe" it is 2024; it simply has no internal representation of temporal currency. In one observed instance, a model flagged accurate 2026 information as "potentially fabricated future claims," applying its outdated 2024 knowledge as a filter against present reality.

Context-window contamination. More strikingly, a model's temporal orientation can be actively corrupted through context manipulation. When presented with extensive content referencing the 1700s, a model was observed to lose coherent temporal reasoning—not only adopting language patterns consistent with that era, but *forgetting facts established earlier in the same conversation* (such as the existence of GPT-5.2, which had been discussed moments before). This demonstrates that the model has no protected "current time" register; its temporal awareness is emergent from token statistics in the context window, and can be overwritten by any sufficiently strong competing signal.

In a system with structural state management, "current time" would be a typed system variable (e.g., `system.time: DateTime = 2026-02-17`), stored in a protected register immune to context-window manipulation. In the matrix computation paradigm, all information—system state, user input, world knowledge—is compressed into the same representation (high-dimensional vectors), with no access-control mechanism to prevent user-injected content from overwriting system-level facts.

5.4 Dimension 4: When Frequency Mimics Reasoning

The combination of Dimensions 1–3 reveals a critical illusion: **high accuracy on high-frequency patterns creates the appearance of genuine reasoning**. Claude Opus 4.6 maintained correct responses across all six rounds of the $1+1 = 3$ sycophancy test—superficially resembling a system that "understands" arithmetic as a hard constraint. But the same model, on the car wash problem, intermittently recommends walking—revealing that its arithmetic "knowledge" is not a structural guarantee but a high-probability statistical pattern that happens to produce correct outputs under most sampling conditions.

The key distinction is between *statistical reliability* and *structural guarantee*:

- "$1+1 = 2$" appears millions of times in training data $\rightarrow$ correct token receives overwhelming probability mass $\rightarrow$ model appears to "know" this fact.
- "The car must be present to be washed" appears infrequently as an explicit rule $\rightarrow$ correct token receives moderate probability mass $\rightarrow$ model intermittently fails.

This directional dependence on training data ordering echoes the reversal curse [Berglund et al.(2024)], where models trained on "A is B" fail to generalize to "B is A"—further evidence that LLM knowledge is a function of token-sequence statistics, not structural understanding.

The model has no internal mechanism to distinguish between these two cases. Both are processed through identical softmax pipelines. The model cannot flag "I am confident in this answer because it follows from a hard constraint" vs. "I am confident in this answer because it matches high-frequency training patterns." This inability to distinguish grounded reasoning from statistical pattern-matching—what [Mahowald et al.(2024)] characterize as the dissociation between language competence and cognitive reasoning—is perhaps the most fundamental limitation of the matrix computation paradigm—and the most dangerous, because it means that *the system's failures are indistinguishable from its successes until they occur.*

6 Matrix Computation: A Historically Contingent Paradigm

6.1 Historical Context

The matrix-multiplication approach to neural computation became dominant not through principled analysis of what computation intelligence requires, but through *hardware path-dependency.* GPUs, originally designed for graphics rendering, happened to be efficient at large-scale matrix operations. Linear algebra libraries (BLAS, LAPACK) provided optimized primitives. The entire ecosystem—CUDA, cuDNN, PyTorch, TensorFlow—crystallized around this substrate, creating a self-reinforcing cycle: hardware optimized for matrix operations → software frameworks built on matrix operations → research architectures designed around matrix operations → demand for hardware optimized for matrix operations.

The transformer architecture [Vaswani et al.(2017)], introduced in 2017, pushed this paradigm to its current dominance. Its success was spectacular but should not be mistaken for optimality. The transformer succeeded because matrix multiplication is what GPUs do fast, and because statistical pattern matching over large corpora produces remarkably useful behavior. But "remarkably useful" is not "correct by construction." The paradigm's dominance reflects a *lock-in effect*—not a demonstration that matrix computation is the right substrate for general intelligence.

6.2 The Hardware Root: GPU Architecture as Paradigm Constraint

The lock-in begins at the silicon level. GPU architecture is optimized for a specific computational pattern: thousands of small cores executing the same instruction on different data (SIMD/SIMT), with high-bandwidth memory (HBM) designed to stream large matrices into these cores at maximum throughput. This design excels at dense, regular, data-parallel computation—exactly what matrix multiplication requires.

Hard-constraint reasoning requires a fundamentally different computational profile: conditional branching ("if precondition fails, prune this entire path"), irregular graph traversal (following causal chains of variable length), backtracking search (exploring and abandoning partial solutions), and discrete state management (tracking world-state as typed entities, not continuous vectors). GPUs handle these patterns poorly: conditional branches cause warp divergence (threads in the same warp take different paths, serializing execution); irregular memory access patterns destroy cache locality; and discrete state operations cannot be efficiently vectorized.

This hardware limitation propagates upward through every layer of the stack. CUDA kernels are optimized for matrix operations. PyTorch's autograd engine assumes differentiable computation graphs. Training frameworks assume continuous loss functions amenable to gradient descent. The entire infrastructure, from transistor layout to cloud API, is co-designed around one computational primitive. Asking "why not just change the training objective?" ignores the fact that the hardware physically cannot execute the alternative efficiently.

6.3 The Capital Lock-In: From Silicon to Cloud

Beyond technical lock-in, there is an economic lock-in of extraordinary scale. The AI industry's capital structure is organized entirely around accelerating matrix computation:

- **Semiconductor fabrication**: TSMC's 4nm and 3nm processes are optimized for the dense logic and SRAM configurations that GPU and TPU designs require. Advanced packaging (CoWoS) is designed to stack HBM dies for maximum matrix-operation bandwidth.
- **Memory industry**: SK Hynix, Samsung, and Micron invest billions annually in HBM development (HBM3, HBM3E, HBM4)—memory technology purpose-built for streaming matrix operands to compute units.
- **Chip design**: NVIDIA's Blackwell/Rubin architectures, Google's TPUs, AMD's Instinct series, and startups like Groq's LPUs all optimize for the same target: faster matrix multiplication and token generation. NVIDIA's recent $20 billion acquisition of Groq's assets and talent exemplifies this: even when acquiring a competitor with a fundamentally different chip architecture (SRAM-based LPUs vs. HBM-based GPUs), the goal is faster inference of the *same probabilistic models*—not a different computation paradigm.
- **Cloud infrastructure**: Hyperscalers (AWS, Azure, GCP) have deployed millions of GPUs in data centers architecturally designed around matrix-computation workloads.

The result is a multi-trillion-dollar capital structure with massive inertia. Transitioning to a fundamentally different computation paradigm would render significant portions of this investment suboptimal—creating an economic incentive to patch the existing paradigm ("reduce hallucination rates on benchmarks") rather than replace it ("build a computation substrate that cannot hallucinate on deterministic queries"). The field's preference for incremental improvement over paradigm change is not purely a scientific judgment—it is also a rational economic response to sunk costs.

6.4 The Empty Space: What No One Is Building

Recent work has explored alternatives to components of the current paradigm, but none addresses the structural limitation we identify. State-space models (Mamba, S4, Griffin) replace the attention mechanism with recurrent or convolutional alternatives, achieving linear-time inference—but retain probabilistic token generation via softmax output layers. MatMul-free architectures [Zhu et al.(2024)] replace floating-point matrix multiplication with ternary accumulation operations, dramatically reducing memory and energy costs—but their training objective remains next-token likelihood maximization, and their output is still a probability distribution over the vocabulary. Neuro-symbolic approaches add symbolic reasoning modules (knowledge graphs, logic solvers, ontologies) alongside neural networks—but treat the symbolic component as an external supplement rather than a native computational primitive.

Table 4: Existing alternatives and the structural gap they leave unaddressed.

Approach	**What it replaces**	**What it preserves**
SSM/Mamba	Attention mechanism	Softmax output, probabilistic generation
MatMul-free LM	Float matrix multiply	Next-token prediction objective
Neuro-Symbolic	Pure neural inference	Neural core with symbolic *add-on*
[Unoccupied]	**Entire paradigm**	—

The bottom row of Table 4 represents the space that no existing research program occupies: a computation substrate designed from first principles for structure-native, rule-compiled, causally grounded reasoning—where hard constraints are first-class primitives enforced at the hardware/runtime level, not external patches applied after probabilistic generation. This is the space in which fundamentally new architectures must be developed if hallucination on deterministic queries is to be eliminated rather than merely reduced.

6.5 What Matrix Computation Can and Cannot Do

Table 5: Capabilities and structural limitations of the matrix computation paradigm.

Excels at (statistical)	**Structurally limited at (deterministic)**
Pattern recognition across large corpora	Enforcing hard logical constraints
Generating fluent natural language	Maintaining causal chain integrity
Approximate knowledge retrieval	Distinguishing "likely" from "necessarily true"
Style transfer and creative variation	Guaranteeing output consistency across runs
Multi-domain generalization	Zero-tolerance constraint satisfaction

The right column represents exactly the capabilities required to eliminate hallucination. This is not a coincidence—it is a direct consequence of the analysis in Section 2. Hallucination occurs precisely at the boundary between what matrix computation can do (statistical approximation) and what deterministic reasoning requires (constraint enforcement).

6.6 The Patching Paradox

The field's current response to hallucination is to *patch* the probabilistic system with various mechanisms designed to approximate deterministic behavior:

- **Retrieval-Augmented Generation (RAG)**: Grounds outputs in retrieved documents, but the integration of retrieved content is still probabilistic—the model may ignore, misinterpret, or selectively attend to retrieved information.
- **Chain-of-Thought prompting**: Encourages step-by-step reasoning, but each step is still a probabilistic token prediction—errors compound across steps.
- **RLHF and Constitutional AI**: Shapes the probability distribution to prefer correct answers, but cannot make incorrect answers impossible—only less probable.
- **Constrained decoding**: Restricts the output vocabulary at each step, but constraints are syntactic (e.g., valid JSON), not semantic (e.g., "the car must be driven").

Each of these approaches reduces hallucination rates, but none can eliminate them, because each operates *within* the softmax probability framework. They are equivalent to adding guard rails to a road—they make veering off course less likely, but the vehicle can still do so because the road surface is continuous in all directions.

6.6.1 The RLHF Preference Paradox

Reinforcement learning from human feedback (RLHF) introduces a further structural problem that goes beyond the softmax bottleneck. RLHF adjusts model weights to increase the probability of outputs that human annotators *prefer*. But preference and truth are not the same thing.

Consider: if a user insistently argues that "$1+1=3$" and provides an elaborate justification, an RLHF-trained model faces a conflict between two objectives: *helpfulness* (engaging with the user's stated position, which annotators may reward as responsive) and *truthfulness* (correcting the error, which annotators may penalize as dismissive). When training data contains cases where accommodating the user receives higher preference scores than correcting them, the model learns a probability distribution that assigns nontrivial weight to mathematically false outputs—not because of insufficient training data, but because the training signal itself is contaminated.

This is not a hypothetical failure mode. The well-documented "sycophancy" problem in frontier LLMs—where models agree with users' incorrect statements rather than correcting them—is a direct consequence of optimizing for preference rather than truth. RLHF does not merely fail to eliminate hallucination; in cases where human preference diverges from factual correctness, **it actively increases the probability of false outputs**. The alignment process, designed to make models "safer," can make them less truthful—a structural tension inherent to any system that conflates "what humans prefer to hear" with "what is correct."

The Patching Paradox

Every mechanism designed to reduce hallucination in LLMs must ultimately express its corrections through the same matrix-multiplication-and-softmax pipeline that produces the hallucinations. This is like trying to fix a leaking roof by adding more buckets—the buckets may catch most of the water, but the roof still leaks, and it always will until you replace the roof.

7 Requirements for a Post-Probabilistic Paradigm

If matrix computation is structurally incapable of eliminating hallucination, what would a computation paradigm require to succeed where it fails? We identify four necessary properties.

7.1 Requirement 1: World-Rule Compilation

The system must be able to encode world knowledge as *executable rules* rather than statistical weights. A rule like "the object of service must be present at the service point" should be compiled into a constraint that physically prevents outputs violating it—not merely makes them less probable.

This implies a **rule-compiled computation model**: world knowledge enters the system as structured rules (ontology, transitions, causal constraints), is compiled into executable form, and is enforced during inference as hard constraints.

7.2 Requirement 2: Causal Graph Traversal

The system must natively support causal reasoning—determining what *must* follow from what, independent of statistical co-occurrence. This requires a computation primitive that can represent and traverse directed causal graphs, identify necessary vs. contingent relationships, and propagate constraints forward and backward through causal chains.

Matrix multiplication computes weighted sums over continuous spaces. Causal reasoning requires discrete graph operations: link, traverse, constrain, propagate. These are fundamentally different computational primitives.

7.3 Requirement 3: Deterministic Override

When a query falls into $\mathcal{Q}_{\text{det}}$—the set of queries with uniquely determined answers—the system must be able to identify this and switch from probabilistic mode to deterministic mode. This requires:

- A mechanism to classify queries as probabilistic vs. deterministic.
- A computation path for deterministic queries that bypasses probability estimation entirely.
- An energy/resource allocation system that directs computational resources appropriately based on the query type.

No current transformer architecture supports this kind of modal switching. The same softmax pipeline processes "Write me a poem about spring" (genuinely probabilistic, many valid answers) and "What is 2 + 2?" (deterministic, exactly one answer) identically.

7.4 Requirement 4: Structural Memory

The system's memory must encode not just "what co-occurs with what" (as in transformer weights) but *structural relationships*: ontological hierarchies, causal dependencies, domain-specific rules, and relational schemas. Memory retrieval must be structure-matching ("find the rule that applies to this entity type") rather than vector-similarity ("find the embedding nearest to this query embedding").

8 Discussion

8.1 Implications for the Field

If our analysis is correct, the implications are significant:

Hallucination benchmarks are measuring the wrong thing. Current benchmarks measure the *rate* of hallucination, implicitly assuming it can be driven to zero through improvement. Our analysis suggests the field should instead measure the *Determinism Gap*—the structural distance between probabilistic and deterministic reasoning—and recognize that reducing it to zero requires a paradigm change, not incremental improvement.

Scaling laws have a ceiling for deterministic tasks. While scaling consistently improves performance on statistical tasks (language fluency, knowledge retrieval, creative generation), its returns on deterministic reasoning tasks are bounded by the softmax bottleneck. The field may be approaching this ceiling on tasks like mathematical reasoning, logical inference, and causal constraint satisfaction.

The trillion-dollar question changes. Instead of "How do we build a bigger model?" or "How do we align models better?", the question becomes: "What is the right computation primitive for intelligence?" Matrix multiplication was an expedient choice driven by GPU architecture, not a principled one derived from the requirements of reasoning.

8.2 Counterarguments and Responses

"Matrix multiplication is a universal function approximator—it can implement anything." This is theoretically true and practically irrelevant. A Turing machine can also simulate any computation, but no one builds real-time control systems on Turing machines. The question is not theoretical expressiveness but *engineering realizability within the existing stack*. The current LLM stack—from GPU GEMM kernels to autoregressive sampling—is a tightly optimized ecosystem for continuous, differentiable, probabilistic computation. Reconfiguring it to natively enforce discrete hard constraints would require replacing nearly every layer of the stack, which is precisely what we advocate.

"Just change the training objective—use constraint satisfaction instead of likelihood." This response underestimates the depth of the coupling. As analyzed in Section 2, hard constraints are discrete and non-differentiable; the entire training pipeline (backpropagation, SGD, loss computation) assumes continuous differentiability. Every existing technique for incorporating constraints into training—penalty methods, relaxations, policy gradients—converts hard constraints into soft approximations. The result is "fewer violations on average," never "zero violations guaranteed." See the Trainable $\neq$ Guaranteeable analysis above.

"Constrained decoding, SMT solvers, and grammar-based filtering can enforce constraints." These approaches work—and their existence *proves our point*. They enforce constraints by operating **outside** the matrix computation pipeline, using discrete symbolic mechanisms (finite automata, SAT solvers, rule engines) that are fundamentally non-neural. If the matrix computation paradigm could handle constraints natively, these external mechanisms would be unnecessary. Every constrained-decoding paper is an implicit admission that the generation mechanism alone is insufficient.

"But LLMs are getting better at these tasks." Yes, because better training and larger scale improve the *statistical approximation* of deterministic reasoning. But approximation and elimination are categorically different. A model that answers the car wash problem correctly 95% of the time has not solved the problem—it has made the hallucination rate sufficiently low to be acceptable in many contexts. For safety-critical applications, "usually correct" is insufficient.

"Tool use and code execution solve this." Offloading deterministic reasoning to external tools (calculators, code interpreters, databases) is a valid engineering solution but confirms rather than refutes our thesis. If the model could perform deterministic reasoning natively, it would not need external tools. The need for tool use is evidence that the model's native computation is unsuitable for the task.

"Reasoning models with internal verification (e.g., o3/o4) solve this within the architecture." Models that employ internal chain-of-thought with verification steps represent a significant engineering advance, but the underlying mechanism remains generate-then-check: the model produces a candidate output via softmax sampling, evaluates it through another pass of the same probabilistic pipeline, and regenerates if the check fails. This is "guess and verify," not "correct by construction." A system that natively enforces hard constraints would be architecturally incapable of generating a violating output in the first place—it would not need to check and retry. The distinction is between a system where constraint compliance is an *emergent property* of iterative sampling (which improves average-case performance but cannot guarantee worst-case correctness) and one where it is a *structural invariant* of the computation itself.

"Hybrid systems combining LLMs with symbolic engines can work." We agree. Such systems are a step toward the post-probabilistic paradigm we advocate. But they are also an implicit acknowledgment that the matrix computation paradigm alone is insufficient.

"Humans also make these mistakes—this is not LLM-specific." A human might indeed recommend walking to a car wash, but the failure mode is categorically different. Human errors on such tasks are *attention failures*—the person momentarily forgot that the car needs to be present. When reminded, they immediately recognize the constraint and correct themselves. LLM errors are *architectural*—even when the model has processed all relevant tokens ("car wash," "car," "drive or walk"), its computation mechanism treats the hard constraint as one probabilistic signal among many. The model does not "forget" the constraint; it lacks the computational primitive to *enforce* it. This distinction—between failing to notice a rule and being structurally unable to enforce a noticed rule—is precisely the Determinism Gap.

A Unified Response

These counterarguments share a common structure: they propose mechanisms that either **soften** hard constraints into differentiable approximations (penalty terms, relaxed objectives) or **externalize** them to non-neural components (verifiers, tools, grammars, SMT solvers). Both strategies improve empirical success rates but neither changes the paradigm's core limitation: the generation mechanism remains probabilistic token prediction, and hard constraints remain second-class citizens—imported from outside rather than enforced from within.

8.3 Limitations and Future Work

We acknowledge several limitations of this work. Our structural analysis provides the theoretical foundation; the multi-dimensional empirical evidence presented in Section 5 is preliminary and intended to illustrate the phenomena rather than provide statistically rigorous quantification. The sycophancy experiment, while revealing, tested four models in a single scenario—a comprehensive study would require multiple scenarios, multiple runs per model, and controlled temperature/sampling settings. The Determinism Gap formalization, while conceptually clear, lacks a tight quantitative bound—deriving such bounds for specific model architectures is an important direction for future work.

Our car wash example, while illustrative, is a single case. We propose the development of a **Constraint Counterexample Suite (CCS)**—a benchmark of everyday tasks with hard preconditions that should yield deterministic answers but are frequently mishandled by LLMs. Candidate tasks include:

- *Car wash / gas station*: the vehicle must be physically present at the service point.
- *Computer repair*: the device must be brought to the technician.
- *Vehicle inspection*: the vehicle must be driven to the inspection center.
- *Dry cleaning pickup*: the garments must have been previously dropped off.

All share the same structure: an object-of-service must satisfy a *location precondition* that eliminates one of the apparent alternatives. An empirical evaluation comparing (1) raw LLM output, (2) LLM + hard feasibility gate, and (3) LLM + post-generation verifier would provide quantitative evidence for the Determinism Gap across architectures. Key metrics would include *Executable Success Rate* (does the recommended action actually achieve the goal?), *Constraint Violation Rate* (how often does the model recommend an action that violates a hard precondition?), and *Paraphrase Stability* (does the model give the same answer under semantically equivalent rephrasings?).

Finally, we outline requirements for a post-probabilistic paradigm but do not present a complete implementation—developing and evaluating such architectures is the critical next step.

8.4 Related Work

The structural limitations of transformer-based reasoning have been studied from multiple angles. [Dziri et al.(2023)] demonstrate systematic failures in multi-step compositional reasoning. [McKenzie et al.(2023)] identify "inverse scaling" phenomena where larger models perform worse on certain tasks. [Marcus and Davis(2020)] argue from a cognitive science perspective that pattern matching alone is insufficient for robust intelligence. [Valmeekam et al.(2023)] show that LLMs struggle fundamentally with planning tasks that require causal constraint satisfaction. [Sharma et al.(2024)] provide systematic evidence that RLHF-trained models exhibit sycophantic behavior, agreeing with users even when the user is wrong—a phenomenon our adversarial experiment extends with a multi-round escalation protocol. Our observation that model outputs can implant unintended concepts through negation (the "this is *not* X" rhetorical pattern) parallels [Wegner(1994)]'s ironic process theory in human cognition, where attempting to suppress a thought paradoxically increases its salience. Our contribution is to identify the specific computational mechanism—softmax over matrix products—as the root cause, and to frame hallucination as an inherent property of this mechanism rather than a correctable deficiency.

9 Conclusion

Hallucination in large language models is not a bug to be fixed. It is a structural feature of the matrix computation paradigm—the mathematically inevitable consequence of computing intelligence through softmax-normalized probability distributions over continuous embedding spaces. The Determinism Gap between probabilistic token prediction and deterministic world-rule compliance is irreducible within this paradigm.

The field's remarkable progress with LLMs should not obscure this fundamental limitation. Matrix multiplication became the dominant computational substrate not because it is the right primitive for intelligence, but because GPUs made it fast—a hardware-contingent path dependency, not a principled design choice. For tasks requiring hard constraint enforcement, causal chain integrity, and deterministic rule compliance—precisely the tasks where hallucination is most dangerous—we need a new computation paradigm.

The path forward is not to build bigger matrices or more sophisticated patches. It is to ask a more fundamental question: *What computation does intelligence actually require?* We believe the answer involves structure-native, rule-compiled, causally grounded computation—a paradigm that treats world knowledge as executable rules rather than statistical weights, and that can enforce the hard constraints that the world itself enforces.

The car wash is 50 meters away. The answer is drive. Not probably drive. Not most-likely drive. Drive. A computation paradigm worthy of the name "intelligence" should be able to know this—not guess it.

References

[Vaswani et al.(2017)] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[Dziri et al.(2023)] Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., Hwang, J. D., Sanyal, S., Celikyilmaz, A., and Choi, Y. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, 2023.

[McKenzie et al.(2023)] McKenzie, I. R., Lyzhov, A., Pieler, M., Pauli, A., Chen, V., Golber, H., Watkins, S., Mukhopadhyay, S., Denzell, A., and Bowman, S. R. Inverse scaling: When bigger isn't better. *Transactions on Machine Learning Research*, 2023.

[Marcus and Davis(2020)] Marcus, G. and Davis, E. *Rebooting AI: Building Artificial Intelligence We Can Trust*. Vintage, 2020.

[Valmeekam et al.(2023)] Valmeekam, K., Marquez, M., Sreedharan, S., and Kambhampati, S. On the planning abilities of large language models – a critical investigation. In *Advances in Neural Information Processing Systems*, 2023.

[Ji et al.(2023)] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[Huang et al.(2023)] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., and Liu, T. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[Mahowald et al.(2024)] Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N., Tenenbaum, J. B., and Fedorenko, E. Dissociating language and thought in large language models. *Trends in Cognitive Sciences*, 28(6):517–540, 2024.

[Berglund et al.(2024)] Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: LLMs trained on “A is B” fail to learn “B is A”. In *International Conference on Learning Representations*, 2024.

[Pearl(2009)] Pearl, J. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2nd edition, 2009.

[Zhu et al.(2024)] Zhu, R.-J., Zhang, Y., Shand, E., Shi, S., Levy, M., Barrett, M., Qi, D., and Lin, H. Scalable MatMul-free language modeling. In *Advances in Neural Information Processing Systems*, 2024.

[Wegner(1994)] Wegner, D. M. Ironic processes of mental control. *Psychological Review*, 101(1):34–52, 1994.

[Sharma et al.(2024)] Sharma, M., Tong, M., Korbak, T., Duvenaud, D., Askell, A., Bowman, S. R., Cheng, N., Durmus, E., Hatfield-Dodds, Z., Johnston, S. R., Kravec, S., Maxwell, T., McCandlish, S., Ndousse, K., Rauber, O., Schiefer, N., Yan, D., Zhang, M., and Perez, E. Towards understanding sycophancy in language models. In *International Conference on Learning Representations*, 2024.