AI as a Team

Bridgewell Advisory LLC

Why AI Behaves the Way It Does

A Practical Explanation of Probabilistic AI Behavior

About This Paper

Bridgewell Advisory LLC is a private AI research and advisory lab focused on the real-world behavior of modern artificial intelligence systems. Our work examines how probabilistic, nondeterministic AI models operate at inference time. Particularly where technical design choices intersect with organizational risk, decision-making, and human accountability.

The content of this paper reflects established properties of contemporary large language models, validated through direct observation, repeated experimentation, and system-level design work across commercial and embedded AI deployments. While these behaviors are well understood in parts of the research community, their implications are often misunderstood or underestimated in practice.

We are sharing this material because AI developers and organizational leaders increasingly rely on these systems in contexts where correctness, uncertainty, and stopping behavior matter more than fluency or completeness. Misinterpreting how these systems generate outputs leads to predictable and avoidable failure modes.

Our objective is straightforward: to help organizations deploy and use AI systems with clearer expectations, better controls, and informed human judgment.

Frank W. Klucznik
1-10-2026

Why AI Behaves the Way It Does

Contents

Why AI Behaves the Way It Does

Introduction

Artificial intelligence systems feel powerful, fast, and increasingly capable. They can summarize complex documents, generate code, analyze data, and engage in fluent conversation. At the same time, they often produce outputs that surprise even technically experienced users. They may sound confident while being wrong, drift away from the original question, or deliver answers that are coherent but subtly misaligned.

These behaviors are often described as hallucinations, bugs, or vendor-specific shortcomings. The term *hallucination* is commonly used, but it is imprecise. A more accurate description is *confabulation*: the generation of plausible, internally coherent outputs in the presence of missing or uncertain information.

This distinction matters because it frames the behavior as a structural property of probabilistic generation rather than a system defect. Viewed this way, what is happening is neither mysterious nor unique to any one company or product. It is a direct consequence of how modern AI systems generate outputs and the conditions under which they are deployed.

A foundational constraint applies to all modern AI systems discussed here: *any system that relies on probabilistic, nondeterministic language models inherits the same class of risk, regardless of vendor, interface, or deployment architecture.* Differences in product design, safety layers, fine-tuning, or system integration can reduce or amplify these risks, but they do not eliminate them.

The central concept in understanding these behaviors is **narrative momentum**: the tendency of a generative system, once it has begun producing a coherent response, to continue extending that response toward completion, resolution, or closure—even when the information required to do so safely is incomplete. This momentum arises because each generated unit conditions the next. As generation proceeds, the system becomes increasingly constrained by its own prior output, and increasingly likely to resolve ambiguity implicitly rather than surface it explicitly.

The purpose of this document is to explain, plainly and from the ground up, why AI systems produce the outputs they do, and to provide a mental model that makes those outputs predictable rather than surprising.

How Generation Works

A common assumption is that an AI system determines a complete answer internally and then outputs it. Modern language models do not work this way. Outputs are generated incrementally, one small unit at a time. Each generated unit conditions the next. There is no fully formed response waiting in the background.

This has several important consequences. Early generation choices disproportionately influence later output. Each step narrows the range of plausible continuations. The

longer generation proceeds, the more committed the system becomes to its own trajectory. This explains why outputs often begin accurately and then drift.

AI systems are often compared to search engines or databases. This comparison is misleading. When an AI system produces output, it is not retrieving an answer and returning it. It is generating a continuation based on patterns learned during training and signals present in the current context. This approach is powerful and flexible. It is also why plausibility can outpace accuracy. An output can sound exactly like the kind of answer that should exist, even when key assumptions are missing, outdated, or incorrect.

Fluency and confidence are properties of this generation process. They are not indicators of correctness.

Determinism, Reproducibility, and Training

It is important to distinguish determinism from reproducibility. Language models are inherently nondeterministic. The same input can yield different outputs because generation is driven by probability distributions, not fixed rules. Techniques such as temperature control, random seeds, caching, and constrained decoding can improve reproducibility—they reduce variance across runs—but they do not change the underlying nature of generation, nor do they eliminate epistemic risk. Reproducible output is not the same as reliable output.

Many discussions conflate training-time improvements with inference-time behavior. The risks described in this document arise primarily at inference time. Even a perfectly trained model must still generate outputs incrementally under uncertainty when deployed. Better training improves the statistical landscape the model operates in. It does not remove the fundamental properties that govern how outputs are produced in real time.

Drift and Narrative Momentum

Drift is often conflated with hallucination or fabrication. In practice, it is usually more subtle. Drift is gradual misalignment between the original intent of a request and the path an output takes over time. It often appears after a correct start. It is incremental rather than sudden, and it can be difficult to detect unless one is actively monitoring for it.

Drift does not require a system failure. It emerges naturally from extended generation under uncertainty. As generation proceeds, the system becomes increasingly constrained by its own prior output. Correct early steps increase confidence, which in turn increases the likelihood that later ambiguity will be resolved implicitly rather than surfaced explicitly.

Longer outputs feel more helpful. They provide detail, explanation, and apparent completeness. Technically, however, longer generation increases risk. Each additional generated unit introduces another opportunity for assumption creep, reinforces earlier choices even if they were slightly off, and increases pressure to resolve ambiguity rather than surface it. Verbosity is not neutral. It is an active risk factor.

Drift should therefore be understood not as a single failure event, but as a trajectory problem: the longer generation continues under uncertainty, the more likely the system is to converge on a plausible, but misaligned, endpoint. The practical question is not whether drift can occur, but how far the system is allowed to run once uncertainty is present.

Why Constraining Output Changes Behavior

Because generation is incremental, controlling how far generation is allowed to proceed changes behavior upstream. When a system is encouraged to stop early, fewer assumptions are introduced, ambiguities remain visible instead of being smoothed over, and the system is less likely to invent connective tissue to force resolution.

This is not about hiding errors by printing less text. It is about preventing those errors from forming by shortening the generation trajectory. Silence, refusal, and requests for clarification are technically meaningful outcomes. They preserve correctness.

Deployment Contexts: Same Risk, Different Visibility

All AI systems that rely on probabilistic language models—whether exposed as chat interfaces, embedded in products, or wrapped in autonomous agents—share the same underlying risk profile: plausible but incorrect output, gradual misalignment over time, premature resolution of ambiguity, and confidence without verification. What varies across implementations is not whether these risks exist, but how visible they are, how quickly they propagate, and whether they can be interrupted.

In **conversational AI systems**, output control has its most direct effect. Limiting output length reduces narrative momentum, surfaces ambiguity earlier, and encourages refusal and clarification. Errors may still occur early, but constrained generation causes the system to fail early and plainly rather than persuasively.

In **retrieval-augmented systems**, retrieval constrains inputs by grounding the model in external sources. This often improves factual alignment, but it does not change how generation itself works. The most common failure mode is false confidence: the presence of retrieved text creates the appearance of grounding while generation still fills gaps implicitly. Output control and refusal conditions remain necessary.

When AI is **embedded in workflows or automation pipelines**, output may not be visible to humans and generated content may trigger downstream actions. The

dominant risk is premature commitment. Output control must be paired with validation layers, human checkpoints, and explicit escalation paths.

In **action-oriented agents**, language output is often transformed directly into execution: API calls, code changes, system actions. The primary risk is not verbosity but premature action. Safe agent design requires hard architectural boundaries: ambiguity must halt execution rather than be resolved automatically, high-impact actions require explicit confirmation, and refusal must be a valid terminal state. Output control is necessary but insufficient; execution control is the real safety mechanism.

Across all contexts, the principle holds: the earlier uncertainty is surfaced, the safer the system becomes.

Inference-Time Control Levers

Modern AI systems expose a limited but meaningful set of inference-time controls. Used correctly, these levers do not eliminate risk, but they materially reduce the likelihood and severity of drift.

Generation length constraints are the most direct control. Hard limits on output length shorten the generation trajectory and reduce narrative momentum. Many failure modes disappear simply because the system is not given enough room to invent connective explanations.

Temperature and sampling controls regulate variance, not correctness. Lower values reduce randomness and make outputs more repeatable, but they do not remove the possibility of confident error. These controls are best understood as stability knobs, not safety mechanisms.

Structured output constraints limit the system's freedom to elaborate. Requiring outputs to conform to schemas, enums, or fixed formats reduces ambiguity, prevents silent assumption expansion, and makes refusal states easier to detect programmatically.

Explicit stopping and refusal conditions should be designed as first-class outcomes: insufficient information, conflicting constraints, or ambiguity should terminate generation rather than trigger resolution. Refusal is not a fallback—it is a correct result under uncertainty.

Prompt constraints that define when generation must stop ("answer only if X is true," "do not infer missing values," "return 'insufficient information' if conditions are unmet") are often more effective than prompts that attempt to guide tone or style.

These levers operate entirely at inference time. They do not depend on better training data, larger models, or architectural complexity. They work because they directly constrain how far probabilistic generation is allowed to proceed.

The Human Role in the System

Humans provide intent, framing, and accountability. They also introduce ambiguity through shorthand, partial corrections, and evolving assumptions. AI systems will faithfully follow these signals unless explicitly constrained.

In practice, "human-in-the-loop" does not mean review after the fact. It means defining where generation must stop and human judgment must resume. When user intent is ambiguous, the correct behavior is escalation, not resolution. When requirements are incomplete, generation should stop. When constraints conflict, AI outputs should surface trade-offs and uncertainty rather than recommendations.

Operationally, human-in-the-loop means designing systems so that stopping, refusing, and escalating are expected behaviors—not exceptional ones.

Why Drift Never Fully Disappears

Drift can re-enter due to structural factors: context decay over time, human shorthand and assumption creep, boundary enforcement fatigue, and confusion between exploration, analysis, and execution. These are properties of extended interaction, not defects.

Conclusion

AI systems produce the outputs they do because of how they generate language: incrementally, probabilistically, and under pressure to continue. The risks observed are not vendor-specific failures, but inherent properties of nondeterministic generation at inference time.

The practical mental model is simple: AI systems are probabilistic generators, not reasoning agents with memory. Control comes from structure and constraints, not intelligence. Short, disciplined exchanges outperform long, fluent ones. Stopping early is often safer than explaining further.

Reliable use comes not from eliminating these properties, but from designing systems and interactions that surface uncertainty early, constrain generation deliberately, and keep humans accountable for decisions and actions.

Predictable systems are more valuable than expressive ones. Constraint is not a limitation—it is what makes dependable use possible.

Appendix A: Operational Guidance and Best Practices for AI Users

Audience: Operators, analysts, product managers, decision-makers, advanced users

Purpose: This appendix explains how to *work with* AI systems safely and effectively, given how they actually generate outputs. It focuses on interaction patterns, not system design.

A.1 What the AI Is Actually Doing (User View)

AI systems do not "know" answers in advance. They generate responses incrementally, one step at a time, based on probability and context. This means:

- Early phrasing matters more than users expect
- Ambiguity is often resolved implicitly unless forced to surface
- Longer answers increase confidence, not correctness

Understanding this helps users recognize when an output is reliable—and when it is merely plausible.

A.2 How to Ask Better Questions

Good results come from **precision**, not verbosity.

Prefer:

- Explicit constraints ("answer only if data supports it")
- Clear scope ("summarize, do not infer")
- Defined stopping rules ("if unknown, say so")

Avoid:

- "You know what I mean"
- Implicit assumptions
- Requests that force resolution ("pick the best option") when trade-offs exist

Remember: the system will try to help even when it should stop—unless you make stopping acceptable.

A.3 Recognizing When the System Should Stop

Stopping early is often the correct outcome.

Signals that should trigger pause or escalation:

- The system introduces assumptions you didn't specify
- Confidence increases without new evidence
- Explanations become longer but not clearer

- The output shifts from analysis to recommendation without being asked

“I don’t know” or “insufficient information” is a **successful result**, not a failure.

A.4 Common User Anti-Patterns

These patterns reliably increase risk:

- **“Just explain more.”**
  This often amplifies narrative momentum and drift.
- **“Try again, but better.”**
  Without changing constraints, this rarely improves correctness.
- **“Pick for me.”**
  This transfers responsibility to a system that cannot own outcomes.
- **Treating fluency as authority.**
  Clear language is not proof of correctness.

A.5 Operational Checklist (Before Acting on Output)

Before relying on an AI output, ask:

1. Did I provide all required information explicitly?
2. Did the system infer anything I didn’t state?
3. Would a short answer have been safer than a long one?
4. Was refusal an available outcome?
5. Am I delegating judgment, or using support?

If any answer is unclear, stop and reassess.

A.6 The User’s Role

Users are not passive recipients. They shape outcomes by:

- Framing questions
- Accepting refusal
- Stopping generation when uncertainty appears

Safe use is a shared responsibility. The system cannot compensate for unclear intent.

Appendix B: Engineering Guidance and Best Practices for AI Developers

Audience: CTOs, platform engineers, LLM engineers, system architects

Purpose: This appendix explains how to design and deploy AI systems that **embody constraints**, surface uncertainty early, and preserve human authority.

B.1 Non-Negotiable Properties of LLMs

All LLM-based systems share these properties:

- Nondeterministic generation
- Probability-driven outputs
- Incremental inference at runtime
- No intrinsic stopping condition

No architecture, vendor, or fine-tuning strategy removes these properties. Design must accommodate them.

B.2 Design for Stopping, Not Completion

Most failures occur because systems are optimized to *finish*.

Best practice:

- Treat refusal as a **terminal success state**
- Make “insufficient information” explicit
- Ensure ambiguity halts progression

If a system cannot stop, it is unsafe by design.

B.3 Inference-Time Control Levers (What Actually Works)

Generation length

- Hard caps (max_tokens, response limits)
- Early stopping conditions
- Most effective drift reducer

Temperature and sampling

- Reduce variance, not error
- Stability control, not safety control

Structured output

- Schemas, enums, fixed formats
- Prevent silent expansion

- Make refusal machine-detectable

Explicit refusal conditions

- Conflicting constraints
- Missing required inputs
- Unresolvable ambiguity

Prompt constraints

- Define when generation must *not* continue
- Often more effective than stylistic guidance

These controls work because they limit how far probabilistic generation can run.

B.4 Deployment Patterns and Required Constraints

Conversational Systems

- Refusal must be first-class
- Long-form answers should be opt-in
- Escalation paths must be visible

Retrieval-Augmented Systems (RAG)

- Retrieval constrains input, not generation
- Must detect source disagreement
- Must halt on retrieval uncertainty
- “Grounded” ≠ safe

Action-Oriented Agents

- Output → action boundary must be explicit
- Ambiguity must halt execution
- High-impact actions require confirmation
- Execution control matters more than verbosity

The most dangerous agents are those that cannot say “stop.”

B.5 Human-in-the-Loop (Engineering Reality)

Human-in-the-loop is not review after execution.

It is:

- Defining decision boundaries
- Enforcing escalation points
- Preventing silent autonomy

Engineering question to ask:

Where must the system stop and a human resume control?

If the answer is unclear, the system is not ready.

B.6 Common Engineering Anti-Patterns

- “RAG makes it safe”
- “More context fixes it”
- “Lower temperature solves correctness”
- “Let the agent decide”
- “We’ll review outputs later”

All of these fail under real-world uncertainty.

B.7 Readiness Questions (Before Deployment)

Every system should answer:

1. What ambiguity is unacceptable?
2. What must halt execution?
3. How is refusal represented?
4. Where does human authority re-enter?
5. What happens when the model is confidently wrong?

If these cannot be answered explicitly, do not deploy.