Introduction

When discussing Agents and LLMs is it useful to recognize 2 distinct frames, epistemic and mechanistic.The **epistemic frame** describes how we interpret, evaluate, and design behavior.The **mechanistic frame** describes what is instantiated computationally.When designing systems that embed LLMs, design begins with epistemic questions of behavior and responsibility, and is subsequently enabled and constrained by mechanistic realities.The affordances of AI as a material bridge these frames.

I. Two Distinct Frames

Design in the Epistemic Frame

Here, design refers to shaping behavior as experienced and interpreted.

It includes:

- Defining role and identity
- Articulating goals
- Specifying tone and boundaries
- Framing knowledge scope
- Setting expectations for capability
- Establishing trust criteria
- Determining acceptable risk

Design questions in this frame look like:

- What should the agent be allowed to say?
- How cautious should it be?
- When should it refuse?
- How should uncertainty be communicated?
- What counts as success?

The object of design is meaning and interaction.

Implications:

- Responsibility lies in normative decisions.
- Failures are epistemic or ethical.
- “Improvement” means reshaping interpretation and expectation.
- Trust and reliability are judged relative to external standards.

Design in the Mechanistic Frame

Here, design refers to the construction and configuration of computational systems.

It includes:

- Model architecture and training procedure
- Fine-tuning or alignment methods
- Context assembly logic
- Retrieval pipelines
- Tool schemas
- Orchestration loops
- Decoding strategy
- Evaluation harnesses
- Deployment infrastructure

Design questions in this frame look like:

- How are logits transformed?
- How is context constructed?
- What constraints are applied to decoding?
- How are tool calls parsed and executed?
- What metrics are computed?

The object of design is the system's mechanics.

Implications:

- Responsibility lies in engineering choices.
- Failures are architectural or procedural.
- "Improvement" means altering model weights, control flow, or constraints.
- Determinism and reproducibility are system properties.

The affordances of AI as a material bridge the epistemic and mechanistic.

Design Patterns vs. Mechanistic Reality

Designers often organize prompts into modular components: system messages, user templates, few-shot examples, retrieval results, tool schemas, skills. These organizational patterns serve legitimate design purposes—consistency, reuse, testing, version control. Mechanistically, these distinctions disappear. The model receives:

> *[system tokens][user tokens][retrieved tokens][tool tokens][previous tokens]...*

A single sequence. There are no boundaries, no labeled components from the model's perspective.

- What designers call "a system message" is mechanistically: tokens serialized at a particular position in the context window.
- What designers call "few-shot examples" is mechanistically: token patterns that condition the probability distribution.
- What designers call "retrieval augmentation" is mechanistically: additional tokens inserted at runtime.

The design abstraction (modular prompt components) is valuable for human organization and iteration. But it is not a mechanistic property of how the model processes input. The model sees one token stream.

The design affordance is “context”, the affordance supports the design of modular prompt patterns.

- Material = the agent
- Affordance = context (what designers can manipulate about that material)
- Design patterns = how designers organize their manipulation of that affordance

LLM Epistemics

[forthcoming]

LLM Mechanistics

LLM behavior emerges from conditional probability distributions over tokens.

At runtime, role, goals, personality, tone — all of it becomes tokens in the context window.

There is no separate internal structure corresponding to “role” or “goal.” Those are design abstractions. Mechanistically:

- The system message is serialized into tokens.
- The user message is serialized into tokens.
- Retrieved documents are serialized into tokens.
- Tool schemas are serialized into tokens.
- Tool outputs are serialized into tokens.

The model receives a single token sequence.

Those tokens condition the next-token probability distribution. That is the entirety of what happens inside the model.

An LLM is a context-conditioned sequence completion engine.

More formally:

- It takes a sequence of tokens (the context).
- It computes a probability distribution over the next token.
- It selects or samples a token according to a decoding strategy*.
- It appends that token to the sequence.
- It repeats until a termination condition is reached.

Decoding Strategies

Decoding strategy is the procedure used at inference time to choose the next token from the probability distribution the model produces.

The model itself computes:

P(token∣context)P(token \mid context)P(token∣context)

This gives a probability for every token in the vocabulary. Decoding determines how one token is selected from that distribution.

The weights are fixed. The distribution is computed. Decoding is the selection policy.

1. Greedy Decoding (Argmax)

Mechanism: Select the token with the highest probability.

token=argmaxP(token∣context)token = \arg\max P(token \mid context)token=argmaxP(token∣context)

Properties:

- Deterministic
- No randomness
- Often repetitive
- Can get stuck in locally optimal continuations

This maximizes likelihood at each step but does not guarantee globally optimal sequences.

Important note on determinism:

If model weights, input tokens, decoding rules, and numerical computation are held constant, identical outputs will be produced across runs. In distributed or heterogeneous environments, floating-point reduction order and hardware differences can introduce small logit variations that may alter token selection in edge cases. Determinism depends on the stability of the execution environment.

---

2. Temperature Scaling

Before sampling, logits are divided by temperature TTT:

Pi=exp(logiti/T)∑jexp(logitj/T)P_i = \frac{\exp(logit_i / T)}{\sum_j \exp(logit_j / T)}Pi=∑jexp(logitj/T)exp(logiti/T)

Effects:

- T<1T < 1T<1: sharper distribution (more confident, conservative)
- T=1T = 1T=1: unchanged
- T>1T > 1T>1: flatter distribution (more diverse, riskier)

Temperature does not change the ranking of tokens. It changes how peaked the distribution is before sampling.

After scaling, sampling is performed.

---

3. Top-k Sampling

Instead of sampling from the entire vocabulary:

- Keep only the top k highest-probability tokens.
- Renormalize.
- Sample from that reduced set.

Controls diversity by limiting tail tokens.

If k = 1, this reduces to greedy decoding.

---

4. Top-p (Nucleus) Sampling

Instead of fixing k:

- Sort tokens by probability.
- Keep the smallest set whose cumulative probability ≥ p.
- Sample from that set.

Example:

- p = 0.9 means “sample from the most probable tokens that together account for 90% of the mass.”

This adapts dynamically depending on distribution shape.

---

5. Stop Conditions

Decoding also includes termination logic:

- Maximum token limit
- Stop sequences
- End-of-sequence token

Without this, generation would continue indefinitely.

---

6. Structured Decoding / Constrained Decoding

In some systems:

- Output must conform to JSON schema.
- Only certain tokens are allowed at certain positions.
- Tool-call formats are enforced.

In this case, decoding is constrained so invalid tokens are masked out.The model still produces probabilities, but the decoding layer modifies which tokens are eligible.

Mechanistic Summary

At each step:

1. Model computes logits.
2. Logits optionally adjusted (temperature).
3. Distribution optionally truncated (top-k / top-p).
4. One token selected.
5. Token appended to context.
6. Repeat.

Decoding strategy is therefore the policy that translates a probability distribution into a concrete token sequence.

It does not change the model's knowledge. It changes how aggressively or conservatively the model explores the distribution.

This is why two calls with identical prompts but different temperatures can yield different outputs — the model distribution is the same; the sampling path differs.

This is why two calls with identical prompts and argmax can yield identical outputs — the model distribution is the same; the sampling path is the same.