Stage 10 → LLM Integration Plan

Context

Stage 10 currently operates only on synthetic ARC-style traces. The parser achieves 100% accuracy on these synthetic runs, using perpendicular energy, smoothing, and matched filtering. However, integration with real LLM hidden states is still pending, as flagged in the project README.

Integration Plan

1. Tap hidden states: Use HuggingFace GPT-2 or GPT-Neo. Extract hidden activations from a mid-layer (e.g., layer 8 of 12). Reduce with PCA to ~d=19, matching Stage 10 spec.
2. Define primitives: Select a small ARC-like set of text transformations (e.g., reverse string, uppercase string, sort characters). For each, collect hidden states during demonstrations to estimate anchors and prototypes.
3. Generate traces: Replace synthetic bumps with real activations. Compute parallel and perpendicular energies from hidden states. Apply smoothing and matched filter as in Stage 10.
4. Parse & execute: Use the parser to decide which primitive fired. Execute the literal transformation on the input string. Compare against the LLM's raw output and ground truth.
5. Metrics: Report accuracy, F1, hallucination, and omission. Compare NGF vs baseline LLM outputs.

Why Stage 10 First?

Stage 10 is the simplest point of entry for LLM integration. It avoids the complexity of Stage 11's Warp → Detect → Denoise pipeline, and focuses on proving that the parser works on real LLM embeddings instead of synthetic signals.

Next step: Draft a new script (e.g., stage10-llm-hook.py) to load GPT-2, extract embeddings, run the geodesic parser, and evaluate NGF vs raw LLM outputs on toy text tasks.