LEE Phase-State Geometry and Logical Mapping

This document captures the **formal logical assignments** and **geometric-phase architecture** underlying the LEE Phase-State Transition Graph. It is directly aligned with LEE's symbolic processing pipeline and serves future runtime design, validation, and visualization goals.

Phase-State Topology: Counterpart Geometry Mode

LEE's internal logic operates across **two conceptual regions**, derived from phase dynamics:

Region A: Constraint-Resolution Zone (Previously called "Counterfactual Space")

- **JAM → MEM**
  - Logic: `⊢ Contradiction ⇒ Archived Resolution`
  - Geometry: Venturi compression — logical closure within minimal passage.
  - Purpose: Jammed logic resolved and persisted for audit or transformation.

Region B: Evaluation-Flow Zone (Previously called "Factual Space")

- **VAC → ALIVE**
  - Logic: `⊢ Axiom ⇒ Initiate`
  - Geometry: Torus expansion — emergent instantiation from vacuum.
  - Codebase: `VAC` states activate via `evaluate_full()` entry point.
- **ALIVE → MEM**
  - Logic: `⊢ Result ⇒ Commit`
  - Codebase: Captured through `State` transition logic, archived post eval.
- **MEM → ALIVE**
  - Logic: `⊢ Recall ⇒ Activation`
  - Codebase: Accessed via memory structure lookup and lambda scope re-entry.

ALIVE → JAM

-
  - Logic: `⊢ Evaluation ⇒ Contradiction`
  - Geometry: Logical collapse.
  - Codebase: Jammed expressions trigger via exceptions or phase overrides.

- **ALIVE → VAC**

  - Logic: `⊢ Simplification ⇒ Vacuum`
  - Exit point for resolved trivial forms or terminal branches.

Transition Matrix (Used in Codebase)

```
python PHASE_TRANSITIONS = { "VAC": ["ALIVE"], "ALIVE": ["MEM",
"JAM", "VAC"], "MEM": ["ALIVE"], "JAM": ["MEM"] }
```

This dictionary is loaded and validated in `evaluation.py`, used to enforce lifecycle invariants.

Logical Mapping of Paths

Transition	Formal Logic Expression	Code Implementation Status
VAC → ALIVE	`⊢ Axiom ⇒ Evaluation`	Implemented (`evaluate_full`)
ALIVE → MEM	`⊢ Result ⇒ Commit`	Implemented (state persistence)
MEM → ALIVE	`⊢ Recall ⇒ Re-execute`	Implemented (scope logic)
ALIVE → JAM	`⊢ Contradiction ⇒ Halt`	Triggered on contradictions
JAM → MEM	`⊢ Resolve ⇒ Archive`	To be structurally tested
ALIVE → VAC	`⊢ Simplify ⇒ Exit`	Planned (future state hook)

Application in LEE Codebase

- Phase states (`VAC`, `ALIVE`, `MEM`, `JAM`) are defined in `core.state.State`.
- Transition validation is embedded in `core.evaluation`, enforced per step.
- Event emission (`LEEEvent`) tracks phase and value pairs during runtime.
- Trace exports support visualization and future GUI integration.
- Geometry-oriented exports (e.g., toroidal or venturi interpretation) remain experimental but grounded in phase transition topology.

Geometric Notes

- **Toroidal expansion**: `VAC → ALIVE → MEM → ALIVE` loops form self-sustaining logical computation arcs.

- **Venturi collapse**: `ALIVE → JAM → MEM` is a narrowing arc, constrictive, restoring coherence through contradiction persistence.
- **No true duality**: These geometries are complementary manifestations of logic's behavior — not metaphysical opposites.

Future Work (Code or Concept)

- Trace hooks to highlight `venturi` vs. `toroidal` flows.
- Assign numeric weights to transitions for runtime entropy metrics.
- Map event patterns into geometric manifolds using real-time visual modules.
- Phase-aware caching and resolution with shape-constrained logic routing.

✳ License and Attribution