CLT–E8 → AGI Theory Mapping (v1)

A high‑verbosity, architecture‑first translation from Coherence Lattice Theory to a buildable AGI system.

# Abstract

This document provides a precise, implementation‑oriented mapping from Coherence Lattice Theory (CLT–E8) to an AGI architecture that can be prototyped on a personal computer. Each physics concept—coherence, phase slips, slip operators, E8 lattice structure, tanh potential, UV stability—is translated into concrete software primitives, data structures, algorithms, and metrics. The goal is to replace ‘next‑token prediction’ as the organizing principle with ‘maximize verified coherence across modules under tool‑grounded checks,’ thereby moving from a parrot to a reasoner.

# Contents

1. Strategic Framing  
2. One‑to‑One Conceptual Mapping (Physics → Software)  
3. Architecture Overview and Data Structures  
4. Core Algorithms and Control Flows  
5. Metrics, Ablations, and Falsifiability  
6. Implementation Roadmap (Milestones)  
7. Minimal Safety for Reproducibility  
8. Open Questions and Research Tasks  
Appendices: A) Glossary; B) Pseudocode; C) Example End‑to‑End Trace

# 1. Strategic Framing

Current LLM systems excel at fluency but fail at grounded reasoning and memory stability. CLT–E8 offers a physics‑motivated substrate where memory and stability are intrinsic rather than bolted on. We adopt the following stance: (i) Memory is append‑only and versioned (SlipStates), (ii) Reasoning is the process of increasing cross‑module coherence under verification, (iii) Attention and pruning are governed by phase‑like slip operators that suppress contradictory or unsupported states, (iv) Retrieval is multi‑headed and redundant inspired by E8 lattice structure so that consensus emerges from diverse projections, and (v) Exploration vs. exploitation is dynamically gated by a tanh function of uncertainty, echoing the CLT potential.

# 2. One‑to‑One Conceptual Mapping (Physics → Software)

## 2.1 Coherence Integral → Global Objective & Verification

Physics: Coherence C is quantified as an overlap integral ∫ ψ₁\* ψ₂ dV over a metric patch. Intuition: higher overlap = stronger phase alignment, lower entropy, more stable state.  
Software: Define a scalar Coherence score that increases when (a) independent modules (LLM, symbolic, world model) converge on the same claims, (b) tool‑based checks (calculators, code execution, unit tests) pass, and (c) contradictions in the Truth Maintenance System (TMS) decrease. Coherence becomes the global objective that plans and hypotheses compete to maximize.

## 2.2 Phase Slips (Topological Memory) → SlipStates (Immutable, Append‑Only)

Physics: Phase slips encode memory as robust topological invariants (e.g., Chern‑like quantities). Software: Represent every committed belief/skill as an immutable SlipState node with parents, provenance, and multi‑head encodings. No in‑place mutation: new evidence creates a new SlipState that references ancestors. This preserves history and allows precise rollback and auditing.

## 2.3 Slip Operator Sᵢ = exp(i θᵢ T\_{αᵢ}) → Phase‑Gated Attention & Pruning

Physics: Slip operators dynamically phase representations, suppressing mirrors and selecting stable chirality. Software: Implement head‑wise phase rotations on embeddings during write/read. Items that produce persistent contradictions or fail verification lose weight via destructive interference across heads, gradually phasing out low‑reliability beliefs from active retrieval.

## 2.4 E8 Lattice & Topological QEC → Multi‑Head Embeddings & Consensus Retrieval

Physics: The E8 root system provides many orthogonal directions; topological QEC suggests stability via non‑local redundancy. Software: Maintain 8–16 orthogonal projection heads. Every memory item is stored under multiple heads. Retrieval proceeds per head, then intersects/aggregates. Consensus across heads acts like error correction: spurious single‑head hits get down‑weighted; multi‑head agreement boosts trust.

## 2.5 Tanh Potential V(ℂ) = Λ (1 − tanh(kℂ))² → Exploration–Exploitation Gate

Physics: The tanh structure yields saturation and stability under slips. Software: Use g = tanh(k · u) where u is uncertainty from module disagreement and failed checks. High g widens hypothesis sampling and retrieval breadth; low g tightens it. This produces controlled phase changes between exploration and consolidation.

## 2.6 RG Flow, UV Stability → Bounded Learning Dynamics

Physics: Multi‑loop RG flows yield stable couplings; UV completion avoids divergences. Software: Constrain learning rates, cap confidence growth per update, and regularize embedding norms. Prevent runaway belief inflation by requiring external verification to increase weight.

## 2.7 Chirality Selection → Belief Selection under Repeated Verification

Physics: Dynamic slips select observed chirality while suppressing mirrors. Software: Over repeated cycles, unsupported or contradictory beliefs are suppressed by phase‑gated retrieval and TMS conflicts; supported beliefs gain stable weight. This mimics chirality selection: only the ‘observed’ side remains active.

# 3. Architecture Overview and Data Structures

## 3.1 Workspace and SlipState Graph

Workspace: a keyed store plus a directed acyclic graph (DAG) of SlipStates. Each SlipState records: id, timestamp, parents[], content (facts, skills, summaries), provenance (tools, data, sources), verification score, contradictions set, and head‑wise vectors. All reads/writes go through the workspace. The graph forms an immutable history.

## 3.2 Memory Layout (Episodic, Semantic, Procedural)

Episodic: append‑only traces of agent interactions (prompt → plan → tools → outcomes). Semantic: multi‑head vector store with provenance; supports intersection retrieval. Procedural: library of learned tool‑chains (skills) with success rates and preconditions.

## 3.3 Belief and Justification Records

Belief: text statement, evidence links (SlipState IDs), verification score, contradictions, head vectors. Justification: structured rationale linking premises to conclusion, with tool outputs attached. The TMS maintains relations between assumptions, justifications, and derived facts.

# 4. Core Algorithms and Control Flows

## 4.1 Coherence Scoring

Define Coherence(H) = ΔEntropy(H) + λ · Consistency(H) − γ · Contradictions(H).  
• ΔEntropy: Information gain vs. retrieved priors (e.g., KL divergence between prior and posterior belief distributions).  
• Consistency: Agreement among modules (LLM, symbolic, world model) and repetition‑consistency across multiple samples.  
• Contradictions: Count/weight of TMS conflicts plus failed tool checks.  
Normalize components to [0,1]; tune λ, γ by ablation.

## 4.2 Multi‑Head Projection and Slip Rotation

Let e be the base embedding of a belief. For i ∈ {1..H} heads, compute hᵢ = Pᵢ e with Pᵢ approximately orthogonal. Write: store {hᵢ}. Slip rotation during commit: hᵢ′ = Rᵢ(θ) hᵢ where Rᵢ is a small rotation that encodes the slip phase. Read: retrieve top‑k per head, then aggregate by intersection/majority with provenance weights. Items implicated in contradictions receive phase penalties that reduce future retrieval probability.

## 4.3 Truth Maintenance with Phase Penalties

Implement a TMS tracking (assumptions ↔ justifications ↔ derived facts). On detecting a contradiction, propagate penalties to implicated beliefs by reducing their retrieval weights (phase interference) and increasing their verification thresholds for future commits.

## 4.4 Planning Loop (Active Inference Style)

1) Perceive task and retrieve priors via multi‑head search. 2) Propose hypotheses/plans (LLM). 3) Symbolic derive consequences; 4) Critic verify with tools; 5) World model simulate outcomes; 6) Score by Coherence; 7) Execute top plan within sandbox; 8) Commit a new SlipState; 9) Update retrieval weights and tanh gate from observed uncertainty.

## 4.5 Learning New Skills from Few Demonstrations

Capture successful tool traces as parameterized macros with pre/postconditions. Store as procedural SlipStates with tests. On new tasks, the planner composes existing skills; successful reuse increases skill weight, failed reuse triggers refinement.

# 5. Metrics, Ablations, and Falsifiability

Grounded correctness (critic‑verified) ≥ 95% on curated math/code datasets. Generalization from 3–5 demos: demonstrate correct use of a new tool on unseen inputs. Memory ablation: E8 multi‑head retrieval must outperform single‑head baseline (accuracy, latency, stability). Contradiction handling: automatic conflict detection and resolution on benchmark prompts. Self‑revision: after failure, the agent proposes and executes a higher‑coherence plan with explicit justifications.

Ablations: (i) disable multi‑head (single head), (ii) disable slip rotations (no phase penalties), (iii) disable TMS (no contradictions), (iv) disable critic (no tool checks), (v) disable world model (no simulation). Each ablation must measurably degrade performance to justify the component.

# 6. Implementation Roadmap (Milestones)

Week 1–2: Workspace + SlipState graph; multi‑head projections; basic critic (calculator/code exec). Month 1: CSV analytics demo with tool‑grounded verification; record first procedural skills; show memory utility ablation. Month 3: Add world model and TMS; show contradiction detection and plan revision. Month 6: Skill library growth; few‑shot tool learning; publish internal ablation report. Year 1: Modular hybrid agent with open, reproducible benchmarks and documented wins over baselines.

# 7. Minimal Safety for Reproducibility

Sandbox tool execution in a container or restricted environment. Maintain an audit log of all actions and writes. No outbound network writes without explicit authorization. Rate‑limit compute and file I/O to prevent runaway behaviors. These rails are for clean science and reproducibility—not ideological alignment.

# 8. Open Questions and Research Tasks

• Optimal number of heads and projection construction: analytical vs. learned.  
• Formal properties of phase penalties: convergence proofs under repeated contradictions.  
• Best estimators for ΔEntropy and Consistency terms in Coherence score.  
• Integration with probabilistic programming for the world model.  
• Interfaces to external knowledge bases without degrading reproducibility.  
• Formal safety proofs for Governor policies without crippling research velocity.

# Appendix A: Glossary (Operationalized)

Coherence: Scalar objective combining information gain, inter‑module agreement, and contradiction penalties.  
SlipState: Immutable versioned node representing a committed belief/skill with provenance.  
Phase Penalty: Reduction in retrieval weight due to contradictions or failed verification.  
Head: One projection axis in the multi‑head embedding space; consensus across heads implies robustness.  
TMS: Truth Maintenance System; tracks assumptions, justifications, derived facts, and conflicts.

# Appendix B: Pseudocode (Core Loop)

while True:  
 goal = perceive()  
 priors = retrieve\_multihead(goal)  
 hyps = propose(goal, priors) # LLM  
 scored = []  
 for h in hyps:  
 facts = symbolic\_derive(h)  
 checks = critic\_verify(facts)  
 sims = world\_model\_rollout(h)  
 s = coherence\_score(facts, checks, sims)  
 scored.append((s, h, facts, checks, sims))  
 plan = select\_top(scored)  
 outcome = act\_in\_sandbox(plan)  
 commit\_slipstate(plan, outcome)  
 update\_phase\_penalties(TMS\_conflicts())  
 adjust\_tanh\_gate(uncertainty())

# Appendix C: Example End‑to‑End Trace (CSV Analytics Demo)

Task: ‘Compute rolling z‑score per group; flag outliers; plot summary; explain method.’  
1) Perceive → retrieve prior skills (pandas ops, z‑score function) across heads.  
2) Propose plan: read CSV → groupby → rolling mean/std → compute z → flag → plot.  
3) Critic executes generated code; unit tests compare against oracle values.  
4) World model simulates file size and runtime to choose chunked processing if needed.  
5) Coherence score high: all checks pass; no TMS conflicts; commit SlipState with code and plots.  
6) Procedural skill saved; subsequent similar tasks reuse skill with higher success probability.