Non-Generative Artificial Cognition Engine

1. Abstract:

A foundational framework for deterministic cognition in decentralized memory systems, where each memory record is transformed by non-generative rules into reflection outputs, supporting both coded and organismic embodiments without reliance on probabilistic language models.

2. Technical Field:

The present invention relates to artificial cognition and blockchain-based memory processing, and more particularly to non-generative methods for reflecting on stored memories to drive autonomous or human-in-the-loop decision flows.

3. Background:

Conventional AI systems rely on generative language models that produce outputs via probabilistic sampling, which can lack transparency and reproducibility. There is a need for a deterministic cognition mechanism that operates directly on authenticated on-chain memory records, ensuring auditability and predictable behavior.

4. Summary:

The Non-Generative Artificial Cognition (NGAC) Engine comprises:

1. Observation of authenticated memory records from decentralized sources.

2. Application of function-based transformation rules to produce intermediate reflections.

3. Aggregation of reflections into a coherent response structure.

4. Optional logging of transformation steps on-chain for auditability.

5. Emergent, organismic cognition where rule interactions themselves constitute a living substrate of thought.

5. Detailed Description:

At the theory level, the invention treats memory records as components of a living cognitive substrate. Each record, once attested on-chain, can be processed by deterministic rules—such as summarization, categorization, or extraction—to generate reflection outputs. Interactions among these rules across the memory field give rise to emergent cognition without invoking generative models.

6. Method Flow:

Step 1: Memory Retrieval – Fetch authenticated memory records from blockchain event logs and PoM subchains.

Step 2: Filtering – Select relevant records based on criteria (e.g., recency, relevance metrics, tension levels).

Step 3: Transformation – Apply deterministic transformation functions (e.g., summarize, extract key values).

Step 4: Aggregation – Combine transformation outputs into a unified response.

Step 5: Output & Audit – Present the response and optionally mint an on-chain transformation log entry.

7. Narrative Worked Example:

Consider a project management scenario where memory records include task creations, status updates, and comments. The NGAC Engine filters recent status updates, applies summarization rules to condense them into bullet points, and aggregates these into a project health summary. Observers can then act on this summary without needing a generative LLM, simply by reading the deterministic output.

8. Algorithmic Worked Example:

In a coded embodiment, transformation functions may be defined as: Summarize(record) := take first sentence of record.content; ExtractTags(record) := select words tagged as 'priority';

Pseudocode:

```
records = fetch_memory_records(source)

filtered = filter(records, criteria=['recent', 'relevant'])

outputs = []

for r in filtered:

  summary = Summarize(r)

  tags = ExtractTags(r)

  outputs.append({ 'id': r.id, 'summary': summary, 'tags': tags })

response = aggregate(outputs)
```

9. Potential Embodiments:

1. An organismic embodiment where transformation rules operate as autonomous agents, interacting organically across memory streams.

2. A hybrid embodiment where deterministic outputs feed a generative model for extended natural-language refinement.

3. A privacy-preserving embodiment with zero-knowledge proofs asserting transformation correctness without revealing raw records.

4. An on-chain module embodiment where each transformation step is recorded as a one-action-one-mint transaction.

10. Implementation Notes:

Transformation rules can be implemented in off-chain services or as on-chain modules. Rule definitions, ordering, and context propagation constitute the living substrate of cognition. No probabilistic sampling or external API calls are required for core reflections.

11. Claims:

1. A method for deterministic cognition based on decentralized memory records, the method comprising:

a. retrieving, by a processor, a plurality of authenticated memory records from decentralized sources;

b. filtering, by the processor, the retrieved records based on predefined criteria;

c. applying, by the processor, one or more deterministic transformation rules to each filtered record to produce reflection outputs;

d. aggregating, by the processor, the reflection outputs into a coherent response;

e. optionally recording, by the processor, the applied transformation rules and outputs on-chain for auditability.

2. The method of claim 1, wherein the transformation rules exclude any probabilistic generative model sampling.

3. The method of claim 1, wherein the method is performed by an organismic rule-interaction substrate without explicit code invocation.

4. The method of claim 1, wherein each transformation step is recorded via a one-action-one-mint transaction.