

# Circulomics: A Case-Based Reasoning Framework for Diagnosing and Resolving Circularity in Logic, AI, and Cognition

*Project report for Computational Models of Cognition CS6251*

Shobhith V. (ED21B069)  
Indian Institute of Technology Madras

November 19, 2025

## Abstract

Circularity appears everywhere: from classical paradoxes (the Liar, Russell) and logical incompleteness (Gödel) to algorithmic fixed points (PageRank, EM), bootstraps in socio-technical systems (credit histories, platform markets), and reflexive cognition (predictive processing, self-modeling). This report presents **Circulomics**, a practical, hybrid Case-Based Reasoning (CBR) system that (1) represents loops as structured cases in a multi-dimensional feature space, (2) retrieves and adapts historical fixes using a tiered similarity mechanism.

## 1 Introduction

Circularity occurs when a system, concept, or reasoning chain loops back onto itself, creating self-referential dependencies that can produce paradoxes, feedback dynamics, or emergent intelligence. Circular reasoning encompasses many paradoxes, puzzles and even scientific breakthroughs. From Zeno's paradoxes to Gödel's incompleteness, circular reasoning has both obstructed and advanced human understanding. In cognition and AI, circularity emerges when systems attempt to model themselves, predict outcomes that include their own actions, or resolve self-referential loops. We see that human cognition has to solve a lot of problems involving circularity like memory, free-will and theory of mind. We also see AI models like LLMs facing limitations that echo classical circular paradoxes (e.g., self-reference, alignment problems, reinforcement instability). By conducting research into this domain we are better able to understand the core ideas behind some of the toughest problems in history.

## 2 Summary

If we remove the mystery from circularity, I discovered some patterns: some loops are *vicious* (provable contradictions, halting/undecidable), some are *benign* (convergent fixed points), and many are *ambivalent* (require grounding or constraints to be useful). Circulomics converts that insight into an engineering pipeline:

1. Formalize what a loop *is* via a compact feature schema,
2. Build a curated case base (100+ annotated cases spanning paradoxes, ML/RL, socio-technical traps, cognitive loops, and temporal paradoxes),
3. Retrieve analogies using a hybrid similarity (veto gates → buckets → weighted similarity + graph kernel)

4. Adapt solutions using short, constrained LLM prompts plus deterministic repair templates,
5. Evaluate solutions with structural checks (type-safety, termination, invariants) and pragmatic tests (simulated rollouts, human rating).

### 3 Motivation and contributions

**Motivation.** Circularity is often treated as an ad-hoc problem in each field. There is value in cross-domain analogies: a parity between the Liar and reward-hacking, or between Russell and self-referential ontologies. Humans are good at analogical transfer; a CBR approach formalizes and leverages that ability. We propose

- A compact, CBR-oriented **feature schema** for representing circularity cases (designed for retrieval and adaptation).
- A hybrid **retrieval** algorithm that combines logical vetoes with graded similarity and a lightweight topological kernel to respect loop structure.
- An **adaptation** protocol that combines deterministic repair templates with LLM prompts for contextualization.
- A rigorous **evaluation plan** (qualitative and quantitative)
- A curated **seed case base** (100 cases) spanning paradoxes, AI/ML, cognition, socio-technical systems and time-travel thought-experiments.

### 4 Design: what to store as a case

For the design of a sucessful CBR system we need cases that are comparable and actionable. Each of our cases contains:

- A Canonical title with a short description.
- Feature vector (structured, categorical and scalar fields capturing the loop anatomy).
- Solutions (2–4 canonical fixes; each fix is a canonical strategy, optionally with a short action recipe).
- Provenance & tags (source references, domain labels).

We have collected our cases from traditional ML, DL and RL algorithms as well as a collection of circularity paradoxes and short stories from the textbook. We also have searched across various policy making and social issues textbooks.

#### 4.1 Feature Schema

The following features define the structural and dynamical “geometry” of circular phenomena. Each feature captures a different aspect of how a loop behaves, breaks down, or can be repaired. Together they form the multi-dimensional representation used in the Circulomics CBR system.

**foundation\_violation** Classifies how the loop violates classical well-foundedness. This is a deep structural property of the circularity. It takes the following arguments:

- **well-founded**: No violation; a grounding base exists and recursion is safe (e.g., convergent fixed-point algorithms).

- **cyclic-ill**: A finite circular dependency that causes contradiction or instability.
- **regress-ill**: An infinite regress with no base case; definitions never bottom-out.
- **impredic-flat**: Self-inclusion or impredicativity; an entity defines itself within its own domain (e.g., Liar, Russell).

**valence\_sign** Describes the polarity of feedback in the loop.

- **positive**: Reinforcing feedback that amplifies a state or error (e.g., filter bubbles, GAN instability).
- **negative**: Self-negating or contradiction-producing feedback (e.g., Liar paradox).
- **neutral**: Feedforward or unidirectional dependency without reinforcement.

**structure\_finiteness** Indicates whether the loop is fundamentally finite or infinite.

- **finite**: A bounded cycle or dependency graph.
- **infinite**: Infinite regress or unbounded chain of dependencies.

**loop\_order** Specifies the “order” or size of the loop.

- Integers  $1, 2, 3, \dots$ : Finite loops of that size (1 = self-loop; 2 = mutual reference).
- **countable\_inf**: A countably infinite regress or hierarchy.
- **continuous\_inf**: Continuous or uncountably infinite loop structure (e.g., iterative optimization landscapes).

**self\_ref\_type** Identifies the type of self-reference:

- **direct**: Explicit reference to itself (“this sentence is false”).
- **indirect**: Encoded or mediated self-reference (Gödel numbering, Yablo sequences).
- **none**: Circularity arises through dependencies, not explicit self-reference.

**topology\_tags** Geometric descriptors of the loop’s shape; used by topology-aware similarity kernels.

- **simple\_cycle**: A single closed loop.
- **nested**: Loops embedded within loops.
- **helix\_regress**: Spiral regress; each step depends on a higher-level copy.
- **mobius\_twist**: Perspective-reversing loop (self/other inversion). For example in social psychology (the spotlight effect), a person would think that other people are judging him/her but other people would be thinking the same and the self/other exchange roles.
- **klein\_bottle**: Boundary collapse; inside/outside distinction breaks down.
- **fractal\_selfsim**: Self-similar recursive structure.
- **figure8\_osc**: Two interacting loops forming oscillatory dynamics. For example in Neuroscience we have inhibitory and excitatory loops. These two loops feed into each other and lead to oscillations.

- **tangled\_hier**: Interleaved or cross-cutting levels (e.g., strange loops in cognition). An example for this is language (grammar rules) where we imagine a clean top down structure where grammar rules sit above everyday usage: rules determine how people speak. In reality however the relationship is interwoven and bidirectional. Everyday usage constantly modifies norms, norms reshape the rules, and those rules then influence how people speak again.

**state\_space** Specifies the nature of the states the system can occupy.

- **finite\_discrete**: Small, discrete sets (true/false, finite automata).
- **countable\_discrete**: Countably infinite states (Turing machines, integers).
- **continuous\_prob**: Continuous or probabilistic spaces (optimization, RL).

**dynamical\_regime** Describes how the loop behaves over time.

- **fixed-attractor**: Converges to a stable point.
- **limit-cycle**: Enters an oscillating cycle.
- **chaotic-strange**: Sensitive and unpredictable (chaotic systems).
- **divergence**: Contradiction or blow-up; no stable state exists.
- **undecidable**: Computationally unsolvable (e.g., Halting Problem).

**hierarchyViolation** Evaluates whether and how definitional or causal levels collapse.

- **flat-impred**: Self-inclusion occurs on a single level (impredicative paradoxes).
- **stratified-pred**: Properly layered; predicative and safe recursion.
- **tangled-strange**: Intertwined levels; mutual causation or strange loops.

**solvability\_modes** Lists the canonical fix families available for this type of circularity.

- **diagonal\_escape**: Move to a meta-level or diagonalize to break self-reference. Example for this is Gödel's sentences such as "This statement is not provable in this system". If you try to evaluate it from within the system you get stuck in self-reference. Gödel proposes the diagonal escape, he reasons about the statement from outside it. At that higher level the statement is true but unprovable.
- **fixedpt\_embrace**: Treat the loop as a fixed-point system and enforce convergence.
- **stratify**: Introduce type layers or separate object/meta levels. Example - In programming languages, types prevent functions from calling themselves in invalid ways (e.g., data level vs. meta level). By keeping levels distinct, self-reference is controlled.
- **approx\_iter**: Stabilize via damping, averaging, annealing, or multi-starts.
- **persp\_flip**: Reverse observer/agent perspective to reinterpret the loop.
- **paraconsist**: Contain contradictions using paraconsistent logic.
- **coinduct**: Accept infinite structures via coinductive invariants.

**benign\_malignant** A continuous rating in  $[-1, 1]$  indicating the harmfulness or usefulness of the loop.

- $-1.0$ : Malignant (catastrophic contradiction, runaway feedback).
- $0$ : Ambivalent or convertible.
- $+1.0$ : Benign or virtuous (productive recursion, stable learning).

This schema is intentionally small: it is sufficient for analogical matches but compact enough for human annotation and learning.

## 5 Retrieve: Hybrid Tiered Similarity Metric

The retrieval engine implements a three-layer hybrid architecture (November 2025) that combines hard vetoes, soft genus bridging, custom categorical matrices, and a novel three-component topology kernel. The metric is specifically engineered for circulomics and achieves near-perfect separation of mathematically incompatible loops while enabling theoretically justified cross-domain analogy (e.g., Russell-type stratification for LLM mode collapse).

### 5.1 Layer 0 — Hard Vetoes (similarity = 0 on mismatch)

- `valence_sign` ∈ {positive, negative, neutral}
- `structure_finiteness` ∈ {finite, infinite}

These are dynamical invariants. Positive vs negative feedback and finite vs infinite structures admit no portable solutions across the boundary.

### 5.2 Layer 1 — Soft Genus Bridge (weight = 0.95)

`foundationViolation` similarity matrix (symmetric):

	well-founded	cyclic-ill	regress-ill	impredic-flat
well-founded	1.00	0.20	0.10	0.00
cyclic-ill	0.20	1.00	0.50	0.70
regress-ill	0.10	0.50	1.00	0.40
impredic-flat	0.00	0.70	0.40	1.00

Key values:  
- impredic-flat cyclic-ill = 0.70: enables Russell fixes for GAN/LLM mode collapse (stratification, teacher-student, unrolling)  
- cyclic-ill regress-ill = 0.50: partial co-inductive overlap  
- well-founded isolated (0.20): induction fails on ill-founded structures

### 5.3 Layer 2 — Weighted Feature Similarities

Overall similarity = weighted average over surviving cases.

### 5.4 Topology Kernel (Novel Component)

50% Jaccard + 30% graph kernel (shortest-path on hand-crafted adjacency) + 20% cosine on 3D embeddings [cyclic strength, oscillatory character, fractal depth]. Yields intuitive clustering (figure8\_osc mobius 0.92, simple\_cycle figure8\_osc 0.85).

### 5.5 Empirical Performance

- LLM self-training hallucination (cyclic-ill, negative, finite, order-2, figure8\_osc, continuous, limit-cycle) → GAN mode collapse at 0.92–0.99 - Classic Liar/Russell → each other >0.90, GAN cases 0.30–0.45 - Novel recursive self-improvement <0.15 → automatic Retain

The metric is the core innovation: mathematically rigorous isolation of incompatible loops with controlled, theoretically justified analogy exactly where solution portability exists.

## 6 Reuse: Direct Solution Adaptation

The Reuse phase is activated when the retrieved case is sufficiently close to the query problem in both structural similarity and textual content, allowing near-direct transfer of stored solutions with minimal transformation.

Feature	Weight	Local Similarity	Rationale (why this weight/-function)
dynamical_regime	0.80	Custom matrix (fixed limit-cycle = 0.60, limit-cycle chaotic = 0.70, fixed divergence = 0.00)	Primary diagnostic for AI/cognitive failures (mode collapse, reward hacking)
state_space	0.80	continuous countable = 0.60, continuous finite = 0.40	Separates DL landscapes from symbolic paradoxes
topology_tags	0.75	50% Jaccard + 30% graph kernel + 20% 3D embedding cosine ([cycle, osc, self-sim] axes)	Geometric intuition; simple_cycle figure8_osc 0.85 even with zero tag overlap
self_ref_type	0.80	direct indirect = 0.70, indirect none = 0.20	Direct needs paraconsistency/-diagonal; none admits game-theoretic fixes
loop_order	0.70	Finite: $1 -  i - j  / \max(i, j)$ ; infinite classes: 1.00	Exact finite order irrelevant (power iteration works for any finite order)
foundationViolation	0.95	Soft matrix above	Genus control
hierarchyViolation	0.40	flat tangled = 0.30	Modulator (Hofstadter vs Russell)
solvability_modes	0.20	Jaccard	Meta; used in Revise
benign_malignant	0.10	$1 -  x - y  / 2$	Outcome only

Table 1: Weights and local similarity functions

## 6.1 Activation Conditions

Reuse is triggered only if **both** conditions are satisfied:

- Structural similarity  $\geq \text{REUSE\_THRESHOLD} = 0.70$
- Fuzzy textual match on the top-1 case  $> 0.50$  (averaged)

Fuzzy textual match is computed as:

$$\text{fuzzy\_score} = \frac{1}{2} \left( \text{name\_sim} + \frac{\text{shared description words}}{\max(\text{words}_q, \text{words}_c)} \right)$$

where name\_sim uses `difflib.SequenceMatcher` ratio and shared words require  $\geq 3$  overlapping terms (case-insensitive).

These dual criteria ensure reuse occurs only for near-identical loop genotypes (e.g., GAN mode collapse  $\leftrightarrow$  LLM self-training collapse routinely scores 0.92–0.99 and passes fuzzy match).

## 6.2 Reuse Prompt Structure

When conditions are met, the system outputs the following paste-ready prompt for the LLM (or human expert):

You are a clever adapter of circular fixes.

Problem: <query.name> - <query.description>

Top match: <top\\_\\_case.name> - <top\\_\\_case.description>

Proposed solutions: <comma-separated top\\_case.solutions>

Solvability modes: <comma-separated top\\_case.solvability\\_modes>

Remix: How can we achieve these solutions for the current problem?

Output:

- Step-by-step description of the adapted strategy (practical, innovative - twist paradox etc.)
- Final 1-sentence executive summary

This prompt deliberately provides **only the single best-matching case** to encourage direct, low-effort adaptation rather than creative synthesis (reserved for Revise).

### 6.3 Post-Reuse Retention (Optional)

After receiving the LLM/human adaptation, the user is prompted:

Retain this adapted solution as new case? (y/n):

On "y", the system appends the query with: - Updated description: original + " [Adapted from <top\_case.name>: <LLM output>]" - Name suffix: " (Adapted-YYYYMMDD-HHMM)"

This creates an evolved case that preserves provenance while enriching the base with refined solutions.

## 7 Revise: Cross-Case Solution Synthesis

The Revise phase is triggered whenever Reuse criteria are not satisfied (max similarity < 0.70 or fuzzy textual match 0.50). It performs controlled cross-genus analogy by feeding the query + top-5 retrieved cases to an LLM for creative adaptation.

### 7.1 Activation Condition

Revise activates in all cases except successful Reuse, i.e.: - max\_similarity < 0.70, or - max\_similarity 0.70 but fuzzy textual match 0.50

This ensures synthesis is used only when direct transfer is theoretically risky or textually distant.

### 7.2 Revise Prompt Structure

The system generates the following paste-ready prompt:

You are a remix master of circular solutions.

Core problem: <query.name> - <query.description>

Top 5 inspirations (pick 1-2 most fitting, ignore order):

1. <case1.name> - <case1.description>. Solutions: <solutions>. Modes: <modes>.
2. <case2.name> - <case2.description>. Solutions: <solutions>. Modes: <modes>.
- ...
5. <case5.name> - <case5.description>. Solutions: <solutions>. Modes: <modes>.

Choose the best 1-2 (e.g., "Top match: #2 for shared dynamics; #4 for solv modes").

Adapt their solutions/modes to this problem - like hacking a paradox fix into an AI workflow.

Output:

1. Chosen cases & why
2. Step-by-step adapted solution description (practical, innovative)
3. 1-sentence final strategy summary

Keep it concise, disruptive: Turn loops into ladders.

Key design features: - Forces explicit selection + justification (traceable reasoning) - Provides only name/description/solutions/modes (no raw features — keeps prompt clean) - "Turn loops into ladders" phrasing encourages breakthrough adaptations

### 7.3 Post-Revise Retention (Optional)

After LLM synthesis, user is prompted:

Retain this revised solution as new case? (y/n):

On "y", the system appends the query with: - Updated description: original + " [Revised from top-<n>: <LLM output>]" - Name suffix: " (Revised-YYYYMMDD-HHMM"

This creates a new hybrid case that captures the creative leap while preserving provenance.

## 8 Retain: Dual-Path Case Base Evolution

The Retain phase implements self-improving growth of the case base through two complementary mechanisms: automatic novelty detection and human-guided evolution after Reuse/Revise.

### 8.1 Activation Pathways

- **Automatic Retain (Novelty Detection):** Triggered when  $\text{max\_similarity} < \text{RETAIN\_THRESHOLD} = 0.10$
- **Manual/Evolved Retain:** Optional after successful Reuse or Revise (user prompted "Retain this ... solution as new case? (y/n)")

### 8.2 Automatic Retain (Novel Circularity)

When  $\text{max\_similarity} < 0.10$ , the system immediately appends the query case with: - Name suffix: " (Novel-YYYYMMDD-HHMM)" - Full original feature vector preserved

This captures genuinely new circular phenomena (e.g., emerging LLM self-contamination modes, recursive self-improvement risks) that fall outside existing coverage.

### 8.3 Manual Retain (Post-Adaption Evolution)

After Reuse or Revise, if user answers "y": - LLM/human output is requested and concatenated into description: " [Adapted/Revised from <source>: <output>]" - Name suffix: " (Adapted-YYYYMMDD-HHMM)" or " (Revised-YYYYMMDD-HHMM)"

Path	Suffix Example
Purpose	
Automatic	(Novel-20251119-1423)
Expand coverage to new loop genera	
Post-Reuse	(Adapted-20251119-1423)
Refine solutions for near-identical cases	
Post-Revise	(Revised-20251119-1423)
Capture creative cross-genus hybrids	

Table 2: Retention suffixes and roles

## 8.4 Implementation Effect

Every interaction either: - Reuses proven fixes (conservative), or - Generates and optionally retains improved/hybrid solutions (progressive)

The case base therefore evolves continuously, turning the CBR system into a living, over time, a comprehensive, self-extending theory of circular reasoning — exactly as intended in the original circulomics vision.

# 9 Project Workflow — From Problem to Solution

## 9.1 Example problem

**Problem:** LLM hallucination (mode collapse in self-training) — a model trained on its own outputs re-ingests errors, producing a reinforcing feedback loop that collapses the output distribution to high-confidence but incorrect modes.

## 9.2 Prompt construction

System / Instruction:

You are a world-class analyst of circularity. Output ONLY a single JSON object with these keys: name, description, solutions, foundationViolation, valenceSign, structureFiniteness, loopOrder, selfRefType, topologyTags, stateSpace, dynamicalRegime, hierarchyViolation, solvabilityModes, benignMalignant.  
Follow the exact value constraints. Think step-by-step internally, but output only JSON.

User / Input:

Problem: LLM hallucination (mode collapse in self-training)

## 9.3 LLM generated feature encoding

```
{"name": "LLM Hallucination (Mode Collapse in Self-Training)",  
 "description": "A model is trained on its own generated outputs so errors are re-ingested as targets, creating a reinforcing feedback loop that reduces output diversity and produces persistent hallucinations. Over iterations the distribution collapses toward high-confidence but incorrect modes.",  
 "foundationViolation": "impredic-flat",  
 "valenceSign": "negative",  
 "structureFiniteness": "finite",  
 "loopOrder": 1,  
 "selfRefType": "direct",  
 "topologyTags": ["simple_cycle"],
```

```

"state_space": "continuous_prob",
"dynamical_regime": "fixed-attractor",
" hierarchyViolation": "flat-impred",
"solvability_modes": ["stratify", "diagonal_escape", "approx_iter"], 
"benign_malignant": -0.8}

```

## 9.4 Remix selection analysis

1. **Primary analogue:** Self-supervised collapse — matches dynamics (optimization on own outputs → trivial attractor).
2. **Secondary analogue:** Russell’s paradox — matches structure (unrestricted self-inclusion; fix by stratification).

## 9.5 Adapted solutions obtained

1. **Stratify data & models.**
  - Define tiers: Type-0 = human GT, Type-1 = model outputs, Type-2 = model-on-model, ...
  - Rule: model may only train on strictly lower-type data.
2. **Stop-gradient firewall on synthetic targets.**
  - Treat synthetic logits as non-authoritative (stop-grad) and scale synthetic loss (e.g.,  $\leq 10\%$  of GT loss).
3. **Inject divergence via a predictor head.**
  - Train a small predictor to forecast the base model; maximize divergence on synthetic samples to create repulsion from trivial modes.
4. **Enforce multi-candidate contrastive separation.**
  - For each prompt, produce  $K$  candidates and enforce contrastive loss to keep representations diverse.
5. **Iterative laddering**
  - At round  $t$ : allow  $M_t$  to train on outputs from  $M_{t-1}$  only; never from  $M_t$  itself.

## 9.6 Summary

Stratify sources, stop-gradient synthetic influence, and force repulsion/diversity — convert a self-referential loop into a directed ladder to prevent mode collapse.

## 10 Limitations

I have analyzed our model and identified its key limitations. Our CBR model works on analogy, and analogy is not proof. A retrieved fix must pass domain-specific invariants to be accepted. Since I am using LLMs for the adaptability, there is always the possibility of them hallucinating actionable steps, this would have to be mitigated via post-checks and human review. There is always the ethical impact of the model to be considered as well. Some problems that I looked at are related to governance and policy making which would directly impact people’s lives so human intervention is needed.

## 11 Conclusion

Circulomics puts circularity at the center, treating loops not as problems to avoid but as real structures I can design and work with. It recognizes that the shape of a loop, its structure and topology matters, and that good solutions come from mixing firm constraints with flexible adaptation. This approach is useful because it makes loops easy to see, easy to reason about, and easier to improve. By clearly representing what the loop is, identifying what can safely be reused, and offering practical ways to repair or reshape it, Circulomics supports both deep analysis and real-world problem-solving.

**Acknowledgements.** I thank Prof. Sutanu Chakraborti for giving us the opportunity to work on this project and the TAs - Adwait and Ratul for their help throughout the course.

## References

- [1] A. Aamodt & E. Plaza (1994). Case-Based Reasoning: Foundational Issues. *AI Communications*.
- [2] D. R. Hofstadter (2007). *I Am a Strange Loop*. Basic Books.
- [3] M. M. Richter & R. O. Weber (2013). *Case-Based Reasoning: A Textbook*. Springer.
- [4] R. S. Sutton & A. G. Barto (2018). *Reinforcement Learning: An Introduction*. MIT Press.
- [5] R. Aharoni (2016). *Circularity: A Common Secret to Paradoxes, Scientific ....* World Scientific.