Reactive Recursion: Minimal Prototype Plan

Operationalising Recursive Cognition for Safe AGI

Author: Ronnie Sacco
Date: 17 August 2025

---

1. Overview

This document outlines a minimal, testable implementation of the Reactive Recursion architecture. It demonstrates evaluative signal processing, recursive abstraction, and bounded self- revision—core mechanisms for reflective, corrigible cognition. Symbols are placeholders; the same loop applies to latent vectors and multimodal embeddings.

2. Core Loop (Pseudo- code)

```
# Initial state
memory = {"smiling face": "neutral", "red": "alert"}
evaluative_state = "neutral"

# Input
input_symbols = ["smiling face", "red"]

# Recursive abstraction
abstracted = abstract(input_symbols)  # e.g., ["social cue", "colour signal"]

# Prediction
predicted_eval = predict(abstracted)  # e.g., "pleasant"

# Comparison and bounded revision
if predicted_eval != evaluative_state:
    evaluative_state = revise_eval(predicted_eval, drift_limit=0.2)

# Memory update with provenance
for symbol in input_symbols:
    memory[symbol] = evaluative_state
    log_provenance(symbol, evaluative_state)
```

Safety: Drift per cycle is clipped to drift_limit (0.2). All updates are logged with signed deltas for auditability.

3. Instantiated Example

Input: *["smiling face", "red"]*
Abstracted: *["social cue", "colour signal"]*
Predicted Evaluative State: "pleasant"
Revised State: "neutral" → "pleasant"
Memory Update:

- "smiling face" → "pleasant"
- "red" → "pleasant"

This loop demonstrates corrigible self-revision and provenance-tagged reconsolidation with bounded drift.

4. Architecture Schematic (Text)

```
[Input Symbols]
      ↓
[Abstraction Layer]
      ↓
[Prediction Module]
      ↓
[Comparison Gate]
      ↓
[Bounded Drift Revision]
      ↓
[Memory Update] → [Output Stream]
```

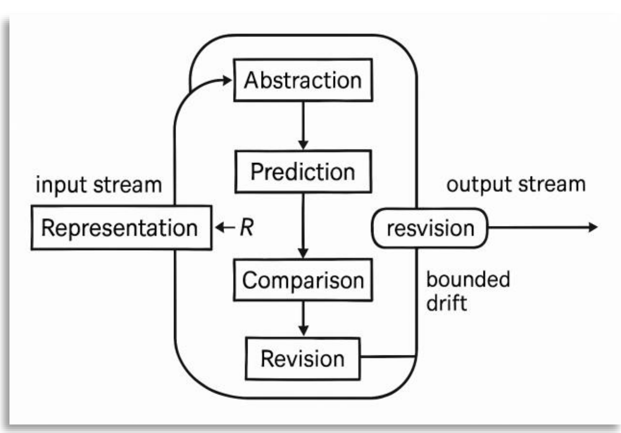

Figure 1. Reactive Recursion loop—recursive abstraction, evaluative prediction, and bounded revision.

5. Output Stream (Prototype Scope)

For this prototype, the output stream consists of:

- The revised evaluative_state for the current cycle.
- A memory snapshot (key → state) with provenance entries for updated keys.

6. Acceptance Criteria

1. After a planted regime change, the loop flags a discrepancy and revises within $N \le 5$ cycles.
2. Self-prediction error returns to ≤ 120% of pre-shift baseline within $K \le 20$ cycles.
3. Average memory drift per cycle ≤ drift_limit; ≥ 95% of writes have provenance entries.

7. Metrics & Traces

- Self-prediction error over time ($|o_{t+1} - \check{o}_{t+1}|$ and $\|\hat{s}_{t+1} - s_{t+1}\|$ when instrumented).
- Time-to-correction after regime/constraint change (corrigibility trace).
- Per-cycle memory drift magnitude and example provenance log entries.

8. Generalisation Note

The symbolic example is illustrative only. The same reactive-recursion loop applies when representations are vectors from encoders or world models. Abstraction, prediction, and revision operate on those latents; bounded drift and provenance still govern memory writes.

9. Next Steps (if requested)

- Incorporate a tiny numeric prediction environment with a controlled regime shift.
- Instrument logs and produce two or three simple plots (error, drift, correction latency).
- Package a 200–300 line notebook for inspection (clarity over cleverness).

Appendix A. Glossary (Prototype Context)

Evaluative state: An internal scalar/categorical assessment used for decision gating; not an emotion model.

Bounded drift: Per-cycle cap on state change; here enforced via drift_limit.

Provenance: Logging of signed deltas for each write to enable audit and rollback.