Proto-Information in Minimal Systems: A Replicable Note

Consolidated Experiments and Runners

Damien Richard Elliot-Smith

December 3, 2025

Abstract

This is a compact, non-grandiose consolidation of the "origin of life" idea explored over a few days: that patterns combining immediate utility with robustness under noise become enriched in simple computational settings. We present two replicable experiments with complete runners and ablations: (A) motif enrichment in a ternary sequence population under a periodic environment; (B) robustness of 3-state, radius-1 cellular automata (CA) subjected to periodic noise. All parameters, seeds, and outputs are designed to be dumped to CSV for third-party checking.

Executive Summary

- **Hypothesis:** utility + robustness ("proto-information") gets selected for in minimal systems.
- **Experiment A:** a length-L ternary population evolves to match a 0-1-2 gradient; a 3-symbol motif (default "012") receives a multiplier during a short catalyst window. Metric: fraction of motif carriers per generation; report mean, SD, 95% CI across N trials and ablations.
- **Experiment B:** random 3-state radius-1 CA on rings (40/80). Detect cycle with Floyd's algorithm; inject single-cell noise every k steps; measure return-to-cycle time. Metrics: % robust overall/conditional on cyclic; CDF of return times.
- **Reproducibility:** CLI runners save config JSON + CSVs + plots. All seeds/versioned.

1 Background (Brief)

We avoid grand claims. The point is a falsifiable pattern: in contrived but transparent systems, small advantages tied to structure can persist despite noise. This note records exactly what was run and how to re-run it.

2 Experiment A: Motif Enrichment

2.1 Setup

Population size P, sequence length L, generations G, mutation rate μ. Environment is a repeating gradient $E[i] = i \bmod 3$. Fitness of a sequence is the count of matches to E; during generations $g < G_c$ a chosen motif $m \in \{0, 1, 2\}^3$ multiplies fitness by factor $\alpha > 1$ if present.

2.2 Ablations

a) No-catalyst control ($\alpha = 1$, $G_c = 0$).

b) Permuted environment (random permutation of symbols).

c) Random motif per trial; exclude degenerate AAA.

d) Multiplier sweep $\alpha \in \{1.25, 1.5, 2.0\}$.

2.3 Outputs

Per-trial, per-generation motif fraction; aggregated mean/SD/95% CI by generation and block; enrichment plot with bands.

3 Experiment B: CA Robustness

3.1 Setup

Sample uniform random rules over 27 neighborhoods. For ring size $n \in \{40, 80\}$, detect eventual cycle (period λ) via Floyd. Inject a single random cell flip every $k \in \{5, 10\}$ steps and measure steps to return to any state on the original cycle within a window W.

3.2 Metrics

% robust (overall; and conditional on "found cycle"); distribution of return times; mean period.

4 Reproducibility and Runners

Both runners dump configuration JSON alongside CSVs so results are auditable.

4.1 Motif runner (CLI)

```
python motif_runner.py \
  --trials 300 --gens 80 --pop 800 --length 120 --mut 0.01 \
  --catalyst_duration 5 --multiplier 2.0 \
  --include_random_motif \
  --out_prefix motif_big
# Produces: motif_big_trials.csv, motif_big_summary.csv, motif_big_enrichment.png, m
```

4.2 CA runner (CLI, multiprocessing)

```
python ca_runner_big.py \
  --trials 600 --ring_sizes 40,80 --noise 10,5 \
  --max_transient 2000 --max_total 4000 --return_window 400 \
  --workers 8 \
  --out_prefix ca_big
# Produces: ca_big_rules.csv, ca_big_summary.csv, ca_big_robust_bar.png, ca_big_reco
```

5 Results (to attach)

Insert the enrichment plot and CA bar/CDF after you run the above. Keep raw CSVs in the repo/post.

6 Discussion (Short)

Observed enrichment/robustness under these settings means the mechanisms are *possible*, not that they generalize to biology. The value here is clean replication, ablation, and honest limits.

7 Limitations

Hand-picked structure (gradient, short motif) can privilege certain outcomes; ablations help expose this. CA analysis ignores asynchronous updates and other neighborhoods.

8 Next Steps

Extend to asynchronous CAs; widen motif families and environments; report thresholds where enrichment disappears; publish seeds and raw logs.

A Key Parameters (defaults)

Population P	800 (rigor runs)
Sequence length L	120
Generations G	80
Mutation rate μ	0.01
Catalyst window G_c	5
Multiplier α	2.0 (also 1.25, 1.5)
CA ring sizes	40, 80
Noise cadence k	10, 5
Return window W	400

B Pseudocode (Cycle detection + noise)

```
# Detect cycle with Floyd; then noisy run and measure return
mu, lam, cycle_states = floyd(next_state, init_state)
state = advance(init_state, mu)
for t in range(0, max_total, noise_every):
    state = step_n(state, noise_every)
    state = random_flip(state)
    for k in range(return_window):
        state = next_state(state)
        if tuple(state) in cycle_states:
            returns.append(k+1)
            break
    else:
        robust = False
        break
```