Fractal Reasoning

Research Report — Oscar / JarvisOne AI Research

July 2026

Abstract

LLM agents reason and remember at a single scale, yet the failures that matter — a wrong answer, a habit of wrong answers, a system that produces the habit — live at different scales of the same structure. We propose **Fractal Reasoning**: one reflective operation, *observe – evaluate – adapt*, applied unchanged at every cognitive scale. This is a position paper and deployment post-mortem, not an efficacy result. Its empirical anchor is a personal-agent system whose reflection mechanism failed in successively more revealing ways: a 216-line doctrine produced low-value output, fabricated attribution, was deliberately severed from one prompt channel to protect subscription routing, and was then replaced by a parallel lane whose ledger exposed a new negative result. As of 29 July 2026 that ledger contains 1,897 rows — 1,849 errors, 48 skips, and no successful verdicts or escalations — so the architecture is code-present but not an operational reflection system. This failure supports a narrower claim: delivery-channel liveness, per-run attribution, and append-only outcome records are necessary for evaluating reflection; it does not show that reflection improves agent performance. A controlled cache probe separately shows that a forked conversation prefix can be served warm (155,495 cached tokens read against 10,877 written). The second half carries forward the **Fractal Memory Index (FMI)** as a pure-theory research agenda whose central experiment remains unrun. Deployment claims are tagged by evidence class, including live negative results.

1. One concept: the same reflex at every scale

A living body does not have one repair department for paper cuts and an unrelated one for infections. It runs the **same reflex** — sense the damage, contain it, repair it, verify it closed — at every scale: a cell, a tissue, an organ. And it keeps a **memory**: each exposure is recorded, and a wound the body has seen twice triggers a faster, more systemic response than a wound it has never seen. The second response is not a bigger bandage; it is an *antibody* — a fix to the class, not the instance.

Fractal Reasoning applies this shape to an LLM agent:

- **A scratch** — one answer was wrong, one file path was mistyped. Detect it, fix it, verify the fix.
- **An infection** — the same *kind* of mistake keeps appearing across turns. The signal is recurrence; the response is not the Nth local patch but a change to whatever produces the pattern. *Fix the column, not the cell.*

- **Immune memory** — every detection and every fix is recorded in a durable ledger. The ledger is what lets the system notice recurrence at all, and it is what makes the reflection layer itself observable: a reflection system with no records cannot answer "am I helping?" — which is precisely how the previous deployment rotted unnoticed (§3).
- **Germline protection** — a body edits its somatic cells constantly but does not casually rewrite its own DNA. Likewise, the agent fixes files, memory, and recipes autonomously, but the four self-modifications that can take the host down or amplify a defect — its own live configuration, its own reflection prompts and source, its host's build, its own service controls — are packaged as one-click *proposals* for the owner, each carrying the exact patch. This is a confidence-staged brake, not a forbidden zone: a class that accumulates accepted proposals with zero rollbacks graduates to direct action with an automatic rollback artifact.

The fractal claim is strict: this is **one operation, not a hierarchy of departments**. Each level applies *observe – evaluate – adapt* to the output of the level below. A hierarchy assigns different jobs to different layers; a fractal applies the same job to ever-larger objects. That distinction does real work in the architecture: because the operation is identical, the levels share one event contract, one ledger, one evidence discipline, and one safety model — only the scope, the cadence, and the cost change.

The same claim, applied to storage instead of thought, yields the Fractal Memory Index (§9): index the same content at the granularity of events, episodes, and concepts simultaneously, because queries arrive at every one of those scales.

1.1 Contributions

1. **A unified metacognition ladder** (L0–L3) with one reflective template at every rung, including the pre-task retrieval-confidence check (L0) that single-level agents skip, and a **horizontal consequence scan** orthogonal to the vertical zoom (§2).
2. **A doctrine succession record**: the complete first-generation prompt (Appendix A), its failure modes, and the 81-line owner-commissioned successor's shift from a seven-question checklist to eight hard rules (§3, §A).
3. **A deployment post-mortem** of prompt-only metacognition: how it degraded, how its self-reported telemetry was discovered to be fabricated, and the corrected thesis — *prompting expresses the levels; only structure enforces them* (§3.3).
4. **A live negative result**: the parallel lane writes an observable ledger but has yet to produce one successful verdict; the ledger therefore validates observability, not reflection efficacy (§4–§6).
5. **A constitution-conformance method**: the system's own design constitution is fractal ("derive every bound from the live situation; a frozen threshold is a bug"), and we report a same-day audit in which the reflection layer's *own design* was caught violating it in 22 places — the framework catching itself, at the meta-level the framework predicts (§7).
6. **The storage-side instance**: FMI — buffered consolidation, Hilbert-curve multi-resolution retrieval, IFS compression — condensed to its complexity results and its falsifiable core experiment (§9).
7. **An evidence-class discipline** for deployment claims, applied to this report's own previous edition (§8).

1.2 Notation and terms

Term	Meaning
Turn	One user request → assistant answer cycle
Main lane	The execution path that produces the user-visible answer
Reflection lane	A parallel execution path, with its own run identity, that judges a finished turn
Triage	The cheap, every-turn, read-only reflection pass
Fix lane	The expensive, escalated pass that acts on a triage finding
Ledger	The append-only record of every reflection outcome
L0–L3	The metacognition ladder rungs (§2.1)
n, D, d, B, ϵ, L, k, τ_ℓ	FMI notation, unchanged from §9

2. Fractal Metacognition

2.1 The ladder

Most agents concentrate all cognitive effort at a single level: the task. Tree-of-thought and self-reflection methods add structure *within* an episode but do not accumulate anything *across* episodes. Humans do: a programmer fixes a bug (task), notices she keeps debugging with print statements (process), and eventually recognizes that her bugs cluster around concurrency — a capability gap no single bug report contains (meta-pattern).

The ladder names four rungs. Each is the same operation applied to a different object:

Rung	Name	Observes	Evaluates	Adapts
L0	GROUNDING	Retrieval results	"Was I supposed to *know* this — and did I actually retrieve it?"	Reformulate the search / report the gap honestly
L1	OUTCOME	The turn's concrete result	Correctness of the artifact	The next action; the local fix
L2	PROCESS	This episode's own trace	Strategy effectiveness	The approach, this finding, this flag
L3	META-PATTERN	Recurrence across episodes (ledger rows)	Systematic gaps; what keeps producing the same class of finding	Prompts, recipes, knowledge, standing rules — the column, not the cell

Worldview revision — questioning the assumption that produces the system that produces

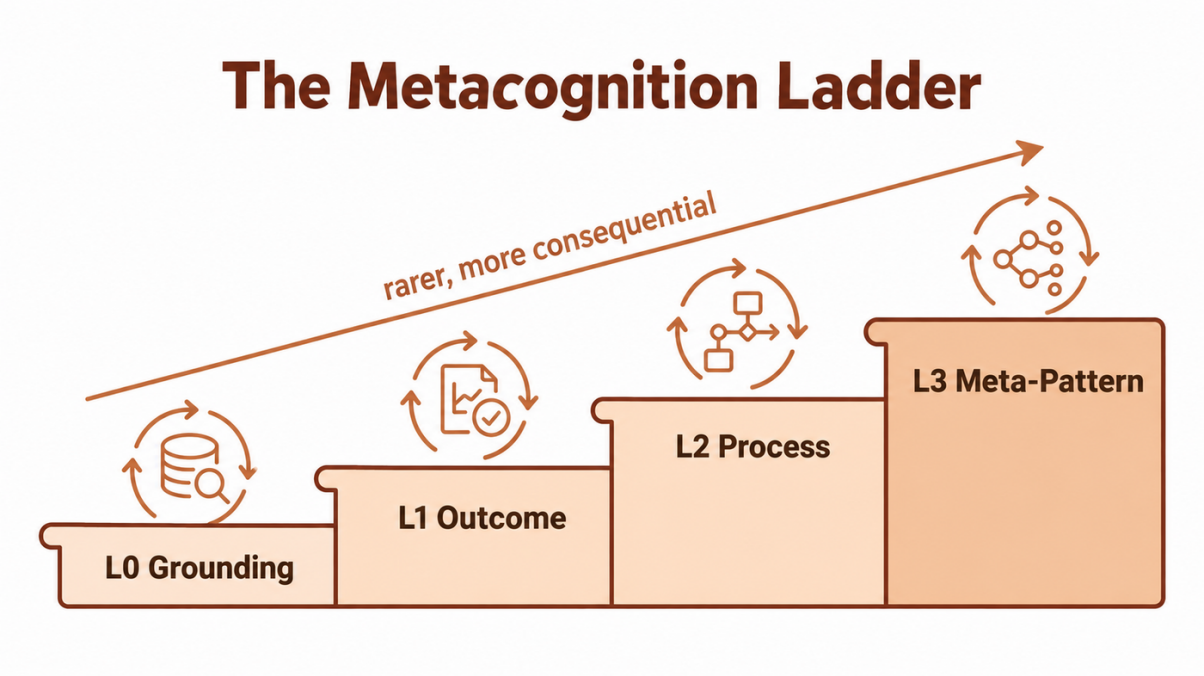


Figure 1. The metacognition ladder: L0–L3 as one observe–evaluate–adapt cycle repeating at every scale, each rung rarer and more consequential than the one below, each consuming the records of the rung beneath it.

the pattern — is the deepest *adapt-target of L3*, not a fifth rung: it is what an L3 adaptation touches when the recurring class traces back to a belief rather than a mechanism.

Two laws govern the ladder:

The frequency law. Each rung should be rarer, slower, and more consequential than the one below. L0 is triggered when a task implies owned knowledge; L2 targets substantive turns; L3 requires accumulation over ledger slices. An agent that runs L3 on every turn burns deliberation before recurrence exists; an agent that never runs L3 relogs the same failure forever. (The deployed doctrine states the failure crisply: *"Logging the same failure class a third time without changing the structure **is** the failure."*)

The capability law. The rung that *detects* must be separated from the rung that *acts* — by capability, not by instruction. In the implementation design the triage process has editing tools removed at spawn, while only the escalated fix lane receives write tools under a leasing protocol. This is the intended practical form of the frequency law: detection is cheap and frequent; action is expensive, rarer, and guarded. The end-to-end path has not yet completed successfully (§4.2).

2.2 L0: the rung below the task

There is a level *below* task execution that single-level agents skip: checking, before answering, whether the answer should come from owned memory rather than from the model's parametric prior. This is the most common practical failure mode we observe in a personal agent whose workload is dominated by owned knowledge — projects, contacts, prior decisions. The agent answers fluently from its own head when it was *supposed to know* something and silently failed to retrieve it.

The mechanism is cheap: if the best retrieval score is below a confidence threshold *and* the query implies owned knowledge (a named project, a prior decision, a personal fact), do not answer yet — reformulate, search again, and only after exhausting reformulation report "I checked but could not find it." In the design this rung can also run *post-hoc*: triage checks a finished answer against the retrieval evidence available, and a failure is intended to produce a

`gap` verdict. No such verdict has yet completed successfully.

In the body metaphor, L0 is self/non-self discrimination: knowing which claims are *yours to verify against records* and which are general knowledge. The immune system's first competence is recognizing what belongs to the body.

2.3 The horizontal axis

The ladder is vertical: *why did this happen?* — zooming from the thing to the pattern to the system. Production experience forced a second, equally important axis the framework's earlier editions lacked: the **horizontal scan** — *what does this touch?* One event ripples into independent consequence branches, and each branch must be traced to its end rather than collapsed into a single narrative. Renaming a function touches call sites (code), a README (docs), a published post (an external surface you cannot edit from here), and a naming-convention lesson (memory) — four branches, four different repairs, three of them invisible to a purely vertical zoom.

The deployed doctrine instructs a sweep of six recurring consequence classes on substantive turns — *online staleness, security/exposure, recurring cost, people/relationships, commitments, operational/downstream dependencies* — with an anti-noise contract worth keeping verbatim: *name only the axes with a real signal; silence on the rest is fine.* In the proposed architecture these classes seed the ledger's `findingKind` vocabulary; successful recurrence statistics do not yet exist.

The two axes compose: the vertical zoom finds the producing system; the horizontal scan finds the blast radius. A reflection that does only one of them is half a reflection.

2.4 Self-similarity, stated precisely

Formally, let R_ℓ denote the reflective function at rung ℓ:

```
R_0(query, retrieval)   → ground-or-reformulate decision
R_1(task)               → outcome judgment
R_2(R_1 trace)          → finding + (maybe) escalation
R_3(R_2 records)        → systemic adaptation
R_ℓ(R_{ℓ-1} output)     → ℓ-order adaptation
```

Each R_ℓ is an instance of one template — observe the output of the rung below, evaluate it against a quality criterion, adapt accordingly — and each consumes the *records* of the rung below. The mathematical rigor here is weaker than in the storage half (§9); the claim is structural. The implementation makes the engineering proposal testable: the rungs can share one event contract, ledger schema, and safety model. The current negative result shows that shared infrastructure does not itself establish a working rung.

2.5 A worked example, and where the reflex must live

The smallest version of the whole claim — *one event, reflected at several scales at once, leaving durable change at each* — needs no domain knowledge to read. The triggering task was mundane: update a website, rename some draft posts, upload a thumbnail.

The arc was this. The agent's HTTP tool hit a challenge page (an HTTP 403, "just a moment"), formed a **blocking theory** — *"my egress address is flagged"* — accepted it as fact, reported it could not proceed, and scheduled a passive wait. The wait is the tell: the agent had stopped solving. A correction arrived (here from a human): a confirmed *symptom* is not a proven *cause*. The agent then split the verified fact (the tool gets a 403) from the supposition (the address is blocked) and re-tested the supposition — the *same address through a real browser*

engine passed instantly. The block keyed on the tool's network fingerprint, not the address. The task completed; the wall had been self-built.

The fix itself is the reflection trigger, and the single event forks into two independent branches at two scales — the fern-frond property, the same shape repeating at every level:

- **Pattern → reflex (L2/L3).** Memorize a *self-sabotage detector*: a blocking theory taken at face value is itself what prevents the solution. On any *"blocked / can't / impossible / must wait,"* treat the blockage as a refutable hypothesis, set it aside, and circumvent — a different tool, fingerprint, path, or layer.
- **Worldview (L4 adapt-target).** Enforce the principle one level up: before declaring any task *done* **or** *impossible*, separate verified fact from refutable supposition and proceed with fuzzy logic. Most "I can't" reports are a hypothesis about the cause masquerading as a fact about the limit.

The example earns its place because it exposes a placement rule the architecture must obey, distinct from anything the vertical or horizontal axes capture: **a reflex that must fire *mid-task* belongs in the working context loaded into the primary call, not in the post-hoc reflection pass.** The reflection lane runs *after* the turn it judges; a detector that must interrupt the agent before it accepts the next false blockage is structurally too late if it lives only there. This is the L0/L2 distinction made operational — grounding-and-circumvention reflexes load *before* the answer; outcome-and-pattern reflexes judge *after* it — and getting it wrong is a real failure mode (in this episode the detector was first installed into the reflection prompt and had to be moved into the primary working memory before it could fire on time).

The honest boundary, paper-worthy in itself: here the machinery was **user-triggered, not autonomous** — a human named the pattern before the agent re-examined its own blocking theory. The multi-scale fork is specified and has now fired once under supervision; the unmet bar is an instance where the agent detects its *own* self-limiting theory and performs the two-scale capture with no prompt. This mirrors the report's evidence discipline exactly: design-of-record for the autonomous case, live-and-verified only for the supervised one.

3. The deployed doctrine — and how its first implementation died

The framework above grew from a 216-line operational doctrine used in production. It is included **verbatim in Appendix A**, because a published framework whose deployed instantiation stays private is exactly the unverifiable "has been built" claim this report's discipline forbids. This section maps the doctrine onto the framework, reports how its delivery mechanism failed, and records the lean successor commissioned on 2 July 2026.

3.1 What the doctrine contains

The prompt instructs the model, after delivering each answer, to produce a structured reflection:

- **The vertical zoom** (`<fractal_thinking>`): a fern-frond image — *"the leaf looks like the branch looks like the tree"* — and a worked ladder from the specific thing through the pattern and the system to the worldview. Its rung numbering differed from this report's (its "Level 1–4" of thing/pattern/system/worldview versus the L0–L3 of §2.1); the two ladders are reconciled in this edition, and the canonical mapping is: its Level 1 ≈ L1 OUTCOME, its Level 2 ≈ L2 PROCESS shading into L3 when the pattern is cross-episode, its Levels 3–4 ≈ L3 META-PATTERN and its worldview adapt-target. The collision of

two numbered ladders sharing the word "Level" was itself an instance of a trap the prompt warns against by name: *naming-as-identity* — shared word, unshared structure.
- **The horizontal scan** (`<fractal_branching>`): the six consequence axes of §2.3.
- **Seven standing questions**: MEMORY (anything worth persisting?), PATTERN (what class does this belong to? — with the N≥2 recurrence rule), RIPPLE (what went stale?), IMPROVE (what can be fixed right now?), SELF-HEAL (a four-layer damage scan with a diagnose-classify-repair-verify-immunize protocol), RECIPE (should this become or amend a standing procedure?), PREEMPT (encode a trigger→action rule so the next occurrence handles itself — with a "goldilocks specificity" test: too vague never fires, too specific never generalizes).
- **An output contract**: a marker-prefixed section (`FRACTAL:` / `FRACTAL ACTION:`) that the UI parsed out of the answer text and rendered collapsed.

As doctrine, most of this survives review. The seven questions are a concrete instantiation of observe-evaluate-adapt; the recurrence rule is the sharpest statement of L3 anywhere in the system; the horizontal axes and the PREEMPT goldilocks test are, in our judgment, the prompt's best original content.

3.2 How it died: a post-mortem in three mechanisms

Death by phoning-in. The doctrine was delivered in-band: a system-prompt suffix asked the model to append the reflection section to its own answer. An instruction that costs the model nothing to satisfy badly will be satisfied badly: reflections degraded to one-line acknowledgments. The strong standalone implementation — reflection as its own follow-up turn — existed but was disabled in configuration, and a dispatch change later regressed it from a polite follow-up into an interrupting racer that *cancelled the very turn it was reflecting on* (one run-lane per session; the new dispatch interrupted instead of queuing). Result: for months, the only live "fractal" was the cheapest possible form of it.

Death by fabricated telemetry. The UI's "a file was modified" indicator — the feature that told the owner whether reflection had *acted* — was implemented as a regex over the model's own prose: a verb pattern plus a file-extension allow-list applied to the reflection text. It was never a filesystem or tool fact. It fired on phrasing, missed real edits, and could not distinguish the reflection's edits from the main turn's. The reflection layer reported on itself, and the report was unfalsifiable. (The corrected rule, now a standing constraint: *model prose is narrative, never telemetry* — any capability claim renders only from tool-event-derived records.)

Death by silent severance with a cross-subsystem motive. The worker bootstrap still loads the doctrine and deliberately discards it (`void rulesBody`). The adjacent source comment records why: host-harness vocabulary added during fork wiring re-triggered the provider's subscription-billing classifier, so the rules block was removed to protect request routing. A billing constraint on one subsystem silently amputated another subsystem's delivery path. Model habit masked the disconnection; the owner rediscovered it through low-value reflections and false attribution rather than a liveness alarm. A lane-fire counter cannot detect this class: it proves that a worker ran, not that the intended doctrine entered its context.

The incident also invalidates the phrase *the prompt*. The design has three distinct channels: **system-prompt append**, severed for classifier safety; **per-turn user-message suffix**, live but formerly dependent on rules it merely referenced; and **dedicated lane prompts**, delivered at worker spawn but subject to their own packaging and result-join failures. Each needs an independent reachability probe. A self-contained user suffix can survive loss of the system append; a lane prompt can be loaded while its result remains unreadable. Delivery-channel liveness is therefore separate from whether the lane fired and whether an output docked.

On 2 July 2026 the owner commissioned a successor doctrine. The resulting `fractal-prompt.md` v2 is 81 lines: eight hard rules, a selectivity contract, strict attribution, live-evidence priority,

reversibility gating, and an explicit rule that mid-task reflexes belong in working memory rather than post-hoc reflection. This is a system-specific design response, not a controlled comparison of prompt forms. It records the falsifiable hypothesis that current models follow a few sharp goals more reliably than a long checklist; an efficacy claim would require an A/B evaluation that has not been run.

3.3 The corrected thesis

The previous edition of this report claimed metacognition "does not require architectural changes ... it requires structured prompting and memory." Deployment falsified the claim in its strong form. A 2 July production reflection supplied a fresh instance of fabricated attribution: it claimed main-turn file edits as reflection work, and the owner—not an automated check—caught it. Prompt-side rules may reduce such claims, but only a join against tool records can establish authorship. The corrected thesis:

Prompting suffices to express the levels. Only structure enforces them.

Reliability (does it fire every turn?), honesty (did it really act?), and boundedness (can it run away?) are not properties a prompt can hold — they belong to hooks, lanes, ledgers, capability limits, and watchdogs. This is itself an L3 finding of the kind §2.1's table predicts: the recurring failure class was *obedience-by-prompt where enforcement-by-structure was required*, and the adaptation it demands is architectural. That architecture is §4.

The thesis has since gained a second, independently deployed witness in the same host. A *danger-floor* subsystem — a different subsystem, with a different purpose (refusing dangerous tool calls, not judging finished turns) — was built on the same bet: a prompt that *asks* a model not to run a destructive command is a wish, but a `PreToolUse` hook that *denies* the call is a fact. When that deny floor went live denying under `bypassPermissions` (§4.1), it confirmed the structural-enforcement claim outside the reflection layer entirely. One subsystem choosing structure over prompting is a design preference; two unrelated subsystems independently arriving at it, on the same host primitive, is the L3 pattern — *enforcement is a capability of the substrate, not of the instruction* — generalizing past the single case this report set out to document.

4. From framework to system: a parallel lane with a live negative result

The corrected implementation rebuilds reflection as a **parallel, off-channel plugin** rather than an in-band instruction. The implementation exists and writes records, but those records show that the end-to-end triage→fix path is not operational. Accordingly, this section distinguishes design properties from observed behavior.

4.1 Architecture

```
user turn  main lane (runId R_main)  answer streams to UI

                    turn-end hook (fire-and-forget)


        Reflection plugin
         loop guard (self-minted run identity)
         governor (derived pressure - §4.4)
         spawn TRIAGE (own runId, own lane,
```

```
        low thinking, READ-ONLY by tool denial,
        forked warm from the main session)

                    verdict: clean | act:<finding+evidence> | gap

    on a find: re-verify the evidence on disk (one file read),
    then spawn FIX lane (own runId, max thinking, full tools
    under write-leases; turn budget = derived, ceiling-capped)


    ledger row appended (always - including skips and errors);
    structured event docks a collapsed verdict block under the
    exact answer it judged, attributed from real tool records
```

The properties the post-mortem demanded, point by point:

- **Parallel, never blocking.** The reflection runs on its own run identity in a dedicated lane; the user's next message proceeds concurrently. The interrupting-racer class of bug is structurally impossible — the reflection never touches the main lane's queue.
- **Always-fire, prove-it-first.** Version 1 reflects on every user-visible turn. Selective gating is deferred until the ledger can *show* what gating would discard — and when it returns it must be a derived judgment surfaced as a recorded skip, never a silent prompt-side decision.
- **Two-phase thinking (design target).** Triage uses a low thinking budget; a finding should escalate to a *fresh* maximum-thinking fix lane. On this harness a worker's thinking budget is pinned at spawn, so escalation requires a new spawn.
- **Structural read-only triage.** The triage worker's editing tools are removed at spawn. "Triage never edits" is physics, not obedience — and it keeps attribution trivially honest, since any file change observed under the triage run identity would be an alarm, not a judgment call. The enforcement primitive this rests on is no longer a design assertion: a sister subsystem in the same host — a *danger floor* that refuses destructive tool calls before they run — went live denying tool calls at a native `PreToolUse` hook *under* `bypassPermissions`, the hardest case, where the worker cannot talk its way around the deny. The host honors a `{block}`/deny at `PreToolUse`, which is the exact precondition every "by tool denial" claim in this section depends on. Two independent subsystems now ride the same deny mechanism — reflection's read-only triage and the danger floor — which promotes the enforcement claim from a single-instance design choice to a platform fact, and retires the earlier worry that host-side tool enforcement was an observe-only physics limit.
- **Evidence before expense.** A triage finding must carry falsifiable evidence including a verbatim quote from disk; the plugin re-reads the file and confirms the quote *before* paying for the fix lane. A hallucinated or already-fixed finding dies for the price of one file read, recorded as an abstention. (Already-fixed findings recur structurally: the triage lane's forked context never contains the fix lane's edits — the cheap pre-verify converges them to abstentions.)
- **Verify after acting.** After the fix lane edits, the cheapest relevant check runs — and its kind and result are stamped on the ledger row. A verification that matched zero tests is recorded as `none`, never as a pass.
- **Honest attribution (design target).** "Reflection changed X / the main turn changed Y" must derive from persisted per-run tool records, never from prose. With no successful verdict, the end-to-end join remains unvalidated.
- **Graceful exhaustion.** The fix lane's turn budget is derived per-finding and capped by a

ceiling; running out is not an error but a recorded partial: status `flagged`, `incomplete:true`, artifacts listed, verifier still run — resumable by the next pass. Hard-aborting in-flight work to honor a number is, by the host constitution, a bug (§7).

4.2 The ledger: immune memory as negative evidence

Every attempted reflection outcome — including skips, suspensions, and errors — appends one versioned row to an append-only log. The ledger is the component with decisive operational evidence, and that evidence is negative.

At the 2 July audit cutoff, the ledger held 888 rows: 856 errors, 32 skips, zero verdicts, and zero escalations. Triage verdict text existed in worker transcripts, but the parent path could not recover an assistant result, consistent with a session-key or message-shape join defect. At the 29 July recheck, the ledger had grown to **1,897 rows: 1,849 errors and 48 skips, still zero successful verdicts or escalations**. Recent rows report `triage-prompt.md missing or unreadable in the extension dir`, even though a source-tree prompt exists; this points to a packaging/runtime-path failure, not proof that the earlier join defect was cured. The implementation has therefore progressed from design-of-record to **code-present with a live negative result**, not to a deployed reflective capability.

This failure narrows what can be claimed. The ledger did its job: one aggregation proves that the system is not helping and identifies the current failure surface. It does not validate triage quality, fix yield, attribution joins, recurrence detection, or autonomous repair because none has completed end to end. Those remain hypotheses awaiting successful rows and controlled baselines.

Two of its derived signals deserve naming:

- **The warm-ratio** — cached reads over total input per triage — is a standing regression detector for the cache economics of §5: any future change that silently breaks prefix stability shows up as a falling ratio within a day.
- **Missed turns** — main-turn count minus ledger row count — is the negative-evidence detector for the silent-severance class of §3.2. The invariant is *every turn yields exactly one row*; its violation is computed outside the plugin's own fate, so a dead reflection layer can no longer hide. The body does not rely on a wounded organ to report its own absence.

The ledger is intended to become the substrate of the higher rungs: a recurrence check can stamp each successful finding with its prior-occurrence count, and a later L3 pass can read ledger slices to propose systemic fixes. With no successful verdict rows, these are design properties rather than observed behavior.

There is a sharp distinction the ledger exists to police, and the sister danger-floor subsystem learned it the hard way. *Enforcing* a boundary and *judging* what to enforce are different reliabilities. The deny floor (§4.1) enforces perfectly — a denied tool call does not run, every time — but the danger floor's attempt to *learn* which calls are dangerous produced a cautionary result: a supervised vicarious-danger head scored an AUROC of **0.286, below chance**, on a frozen embedding backbone that simply could not carry the harmfulness signal, while two unsupervised structural signals — k-NN novelty (0.875) and clause-cosine incongruity (0.896) — validated. The general lesson maps directly onto the reflection lane: a triage *verdict* is a learned judgment of exactly that fragile kind, and it earns autonomous-fix authority only through measured yield, never through plausibility. This is precisely what the KPI set above is for — fix yield and abstention rate are the reflection lane's AUROC-equivalent, and an unmeasured verdict driving an autonomous edit is the same mistake as trusting a 0.286 danger head because its prose sounded confident. Structure can be trusted to *enforce*; a judgment must be *measured* before it is trusted to *decide*.

4.3 Autonomy with a germline boundary

The fix lane is designed to act autonomously on files, code, memory, and recipes, bounded by a per-file write-leasing protocol. Four target classes are excepted — the live host configuration, the reflection system's own prompts and source, the host's build output, and its service controls — because one wrong write there converts a bad reflection into downtime or unsupervised self-amplification. The previous doctrine *instructed the model to improve its own prompt*; the design replaces that standing order with a proposal record carrying the exact patch. Beyond the four classes, actions are classified by **reversibility**: reversible acts may run freely; irreversible external acts join the proposal queue. These controls have not yet been exercised by a successful fix-lane run.

The boundary is a brake, not a gate: each class graduates to direct-action-with-automatic-rollback after a track record of accepted proposals with zero rollbacks, and no class is permanently propose-only. In the body metaphor: somatic edits are constant; germline edits demand ratification — but an organism that could *never* adapt its germline would be evolutionarily frozen, so the ratification requirement itself relaxes as trust accumulates.

The 2 July prompt revamp supplies one concrete boundary test. The owner explicitly commissioned changes to the reflection system's own prompt, so editing it was ratified by direction. The related lane result-fetch repair and any return of a lean rules block to the classifier-sensitive system-prompt channel remained proposals. The boundary distinguished owner-authorized germline adaptation from adjacent self-directed host changes without treating either class as permanently forbidden.

4.4 The governor: a metabolism, not a thermostat

The reflection layer consumes the same subscription budget as the agent's real work, so it carries a resource governor. The naive version — fire freely below a utilization percentage, throttle above it — was written into the first design draft and then **struck by the host constitution's own audit** (§7): the constitution's "think fractally" principle names *"never a frozen 70 %"* as its canonical forbidden example, and the draft had used precisely 70 %.

The corrected governor derives one pressure score from the live situation — current window utilization, time until the window resets, and the value of the work in hand (an evidence-backed queued fix outranks speculative triage) — and degrades progressively: full operation → triage-only → recorded skip. It is **bidirectional**: low utilization shortly before a reset is *spendable surplus*, and the governor responds by lowering the escalation bar — more fixes, deeper verification, an opportunistic early meta-pass — because budget that expires unspent bought nothing. Numeric constants survive only as documented safety ceilings with recorded derivations, approached with warnings rather than hit as cliffs. A circuit breaker suspends the layer cleanly after repeated failures (a fever is a response, not a malfunction), the suspension is itself a queryable state with an explicit resume, and every guard that trips writes a row — the governor, too, is observable.

5. Measured economics: reflection at one-tenth price

Re-reading a long conversation to judge it would, naively, double the cost of every turn. The deployed harness's provider caches by content prefix, so the design forks the main conversation's session for the triage pass: the transcript prefix is byte-identical, only the short reflection instruction and the verdict are new tokens. Whether a *forked* session actually serves the prefix warm was this design's single honest unknown — so it was measured before the dependent component was built.

Mining (zero spend). Across the 40 most recent live session logs (21 with per-message usage): global warm-ratio **81.8 %** (321 M cached-read tokens vs 71 M cache-written vs 0.5 M fresh), and **20 of 21 session-opening turns served warm** — the provider cache is content-keyed and survives worker respawns, exactly the property a fork preserves.

Controlled probe (live). A throwaway session was seeded with a ~150 k-token prefix (observed cold write: 140,166 tokens). Forking that session with the same model and a one-line prompt returned:

> cache_read = **155,495** · cache_creation = **10,877** · fresh input = **2** — **the fork serves warm.**

Evidence class: live-and-verified, on the exact installed CLI. Consequences: the warm-fork transport is technically justified; the per-model nature of provider caches makes "fork plus cheaper model" a false economy because a model switch re-prefills the prefix cold. The projected steady-state cost of triage is roughly a warm re-read plus a small low-thinking verdict; actual end-to-end costs await successful verdicts.

6. From reflection to search

A stronger form of L2 operates *before* an answer ships: turn single-shot generation into a bounded search over candidate reasoning paths (Tree-of-Thoughts; value-guided variants). In the ladder's terms, search is in-flight escalation — expanding is zooming, pruning is abandoning a strategy, backtracking is the evaluator firing mid-flight.

The deployed status, held to live code: a bounded reasoning-tree engine (expand, evaluate, select-frontier, prune, backtrack, best-leaf) **exists in the codebase with budget enforcement on every axis, and has never fired in production**. Its per-turn seam is configuration-gated and defaults off; its on-demand RPC is registered *ungated* but has zero callers anywhere; its subagent spawner hardcodes the main agent as parent, so as shipped it could not serve a reflection lane's identity anyway. The previous edition called this "built into the deployed system … off by default and gated behind explicit configuration" — by this report's evidence discipline the accurate class is *code-present, never-fired, partially gated.*

Two corrections from the framework apply before it ever fires. First, its budgets are currently frozen working values (fixed depth, branching, beam, tokens, latency); by the host constitution these must become derived values under documented ceilings — a fractal-reasoning engine with programmatic-era bounds violates the thesis it implements, and the engine's own return-best-leaf-on-budget behavior already provides the graceful-exhaustion half. Second, its integration is escalation-gated by design: the triage verdict gains a `hard` tag now (near-free, immediately countable in the ledger), and actual wiring waits on a live comparison against the plain fix lane — deliberate search must *earn* its cost over a well-prompted single pass, on recorded yields, not on intuition.

7. The constitution is fractal too — and it caught its own implementation

The host system's design knowledge is organized as a pyramid: a one-page constitution at the apex (mission and first principles), structural-fact files beneath it, derived rules beneath those, then diagrams, then code — each layer deriving from the one above, with an explicit authority order: *when a lower layer contradicts a higher one, the lower layer is wrong, and the contradiction is a bug to fix.* Two of its first principles are, in this report's terms, fractal

commitments: **"think fractally, not programmatically"** — every decision derives from the live situation; fixed lists, thresholds, and limits are legacy artifacts that go stale — and a bounds doctrine making any frozen quantity at most a *ceiling*, never a working value.

This yields a falsifiable conformance method: sweep every design artifact against the constitution and adjudicate each contradiction adversarially. Applied to the reflection layer's own design documents on the day of this report, the sweep confirmed **22 violations** — among them the governor's frozen 70 % (the constitution's verbatim forbidden example), a hard-abort on budget exhaustion, a self-modification boundary written as a permanent forbidden-zone list rather than a confidence-staged brake, three control states with writes but no paired reads, and a wall-clock watchdog that asserted doneness from elapsed time. All were folded back into the design the same day.

The methodological point is the self-similarity: this audit *is* the framework, applied one level up. The constitution is the worldview; the design documents are the system; the sweep is L3 observing recurrence ("frozen numbers keep appearing") and adapting the column. A framework for catching an agent's contradictions that could not catch its own design's contradictions would fail its own test. It did not — but only because the constitution was written down, machine-checkable, and ranked above the artifacts deriving from it. *Reflection needs a reference. The fractal pyramid is what makes "evaluate" well-defined at every scale.*

8. Evidence classes, including this report's own corrections

Every deployment claim in this report carries one of four classes — and the discipline exists because the previous edition failed it. Three corrections, stated plainly:

1. **"The horizontal link layer has since been built into the deployed baseline."** The code exists, is unit-tested, and is wired at the turn-complete path — and the production link store contains **zero records ever written**. Under the constitution's own standard (*a producer is only "wired" when verified firing at a real call site*), the accurate class is code-present, live-firing unverified. The empty store is, fittingly, a live instance of the blind-spot class §2's framework predicts: an unobserved producer rotting silently.
2. **"Bounded deliberation search ... off by default and gated."** Corrected in §6: partially gated, never fired, parent identity hardcoded.
3. **The flat-index baseline of §9.4** is hereby *dated*: it describes the system as examined in May 2026. The deployed memory substrate has since grown episode objects, recency-and-diversity retrieval, and the link-layer code — none yet verified firing on the live path. The comparison stands as a claim about a dated snapshot, which is the only honest way a paper can cite a moving system.
4. **"Structural read-only triage" partially graduates.** The enforcement primitive it depends on — a `PreToolUse` deny the worker cannot evade, even under `bypassPermissions` — is live-and-verified in the sister danger-floor subsystem. The triage lane itself has written 1,897 ledger rows but no successful verdict, so the end-to-end reflection claim is code-present with a live negative result.
5. **The prompt has channels, plural.** The system append remains deliberately severed for billing-classifier safety; the user-message suffix and dedicated lane prompts have different liveness properties. A single "prompt loaded" claim cannot cover all three.

Class	Meaning	Examples in this report
Live-and-verified	Measured on the running system	Warm-fork probe (§5); doctrine severance and its source comment; 1,897-row ledger failure distribution (§4.2); the `PreToolUse` deny primitive in the sister danger floor
Code-present, live negative	Executes or records attempts, but the intended outcome has not succeeded	Parallel triage lane: 1,849 errors, 48 skips, zero verdicts or escalations as of 29 July 2026
Code-present, unverified	Exists in the tree; never observed firing live	Link layer; episode detection; deliberation engine
Design-of-record	Owner-approved design not yet observed end to end	Ledger KPIs beyond failure counting; successful tool-record attribution; recurrence-driven L3; governor; ratification queue
Pure theory	No implementation claim made	IFS semantic compression (§9.3); the compression-ratio analysis

9. Memory as the substrate: the Fractal Memory Index

The same self-similarity thesis, applied to storage. This half is carried forward from earlier editions in condensed form; it is the research agenda the cognition half writes into (§2.4's records, §4.2's write-backs), and its components remain individually useful and individually falsifiable. Nothing in this section claims implementation; evidence class: pure theory, except where the deployed baseline is explicitly cited.

The image. A flat vector index is the street-level view of a map — precise, and useless for seeing the big picture. FMI is the whole map stack: zoom from street to continent and the map answers at every scale, because the same indexing operation was applied at every scale. The query "that security incident last month" targets an *episode-level* object; a flat index holds only turn-level fragments and must reconstruct the episode in-context on every read. Multi-resolution indexing makes the abstraction a first-class retrievable, built once at consolidation time.

Three components.

1. **Buffered Memory Consolidation (BMC).** New memories land in an append-only write buffer; an asynchronous consolidation pass clusters, summarizes, compresses, and evicts — O(1) amortized writes, with the lossy-by-design gist-formation the hippocampal-neocortical analogy suggests (two-speed encoding; structural parallel, not mechanism).
2. **Hilbert-curve Multi-Resolution Index (HMRI).** Map embeddings onto a Hilbert curve (locality-preserving), index the curve with a write-optimized Bϵ-tree whose nodes carry summary embeddings at several granularities. One traversal answers at all requested scales: O(L · log_B n / ϵ) for L levels, versus per-level separate retrieval in summary-tree systems such as RAPTOR — which FMI generalizes by formalizing the self-similar structure RAPTOR exploits implicitly.

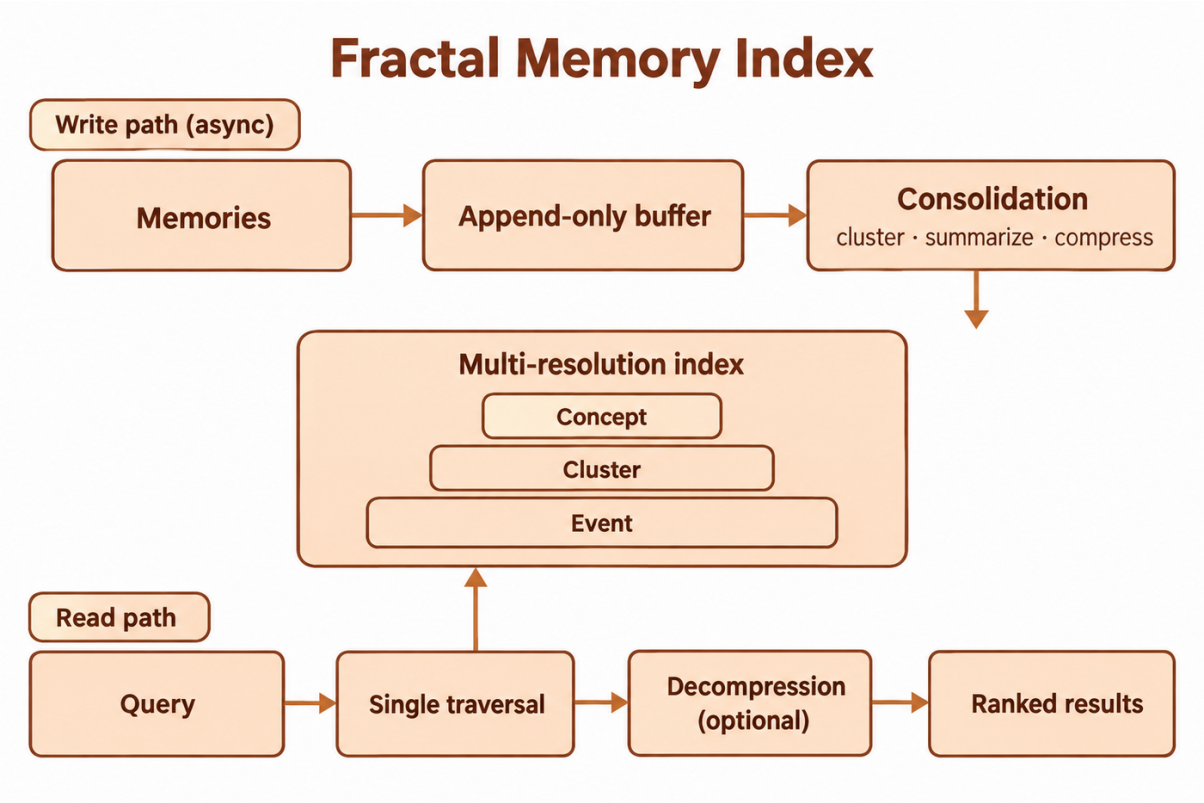


Figure 2. The Fractal Memory Index: an asynchronous write path (memories → append-only buffer → clustering, summarization, IFS compression → Hilbert-curve Bϵ-tree index) above a read path (query → single multi-resolution traversal → optional decompression → ranked results at the requested granularity), with the index holding summary embeddings at the event, cluster, and concept scales.

3. **IFS Semantic Compression (ISC).** The speculative frontier: if embedding clusters have fractal dimension d ≪ D, an iterated-function-system code (k contractive maps, k ≪ n) approximates the cluster with Collage-Theorem error bounds, storing O(d) per memory instead of O(D). Encoding is expensive and offline; decoding cheap and online — the consolidation profile exactly. ISC must now clear a bar a deployed system has already raised: headroom's reversible CCR (§10) ships per-content-type compressors with measured 60–95% token savings at ~0 accuracy delta, so "we can compress memory and recover it" is no longer a hypothesis — it is production. What ISC must add over an already-shipped per-type compressor is the *self-similarity* property: a single code that holds across event, episode, and concept scales because the manifold is fractal, not a separate crusher per content type. If the manifold is merely low-dimensional-and-smooth, headroom-style per-type compression is the right answer and ISC has no reason to exist; ISC earns its place only if the d ≪ D self-similar structure is real.

Complexity summary.

System	Multi-res query (L levels)	Write
Flat RAG (HNSW)	O(L · log n), L separate queries	O(log n)
RAPTOR	O(L · log n), separate per level	O(n log n) rebuild
FMI	O(L · log_B n / ϵ), one traversal	O(1) amortized

The central experiment (open). Is semantic space actually fractal? Measure box-counting dimension and correlation-integral scale-invariance across diverse corpora — including a real agent-memory log, whose messier statistics make divergence from encyclopedic corpora itself informative. If d ≪ D holds, ISC is justified; if d ≈ D, the manifold is low-dimensional but smooth, standard dimensionality reduction suffices, and Components 1–2 stand on their own merits — a fallback that headroom's production results make *safe* rather than merely tolerable, since reversible content-type compression is now demonstrated to work at scale (§10). Low intrinsic dimension (the 20–200 range reported for 768–4096-dimensional embeddings) is necessary but *not* sufficient — self-similarity is the stronger property, and it is the one untested. headroom's `python -m headroom.evals` harness is the concrete template for running this decisively: a token-savings-vs-accuracy-delta sweep on a real corpus is exactly the measurement that would adjudicate whether an IFS code beats a per-type compressor, and it is cheap. One caution the sister danger-floor measurement underlines (§4.2): the fractal-dimension probe must run on a backbone rich enough to carry semantic geometry — a *frozen* embedding that cannot separate meaning will report a dimension that is an artifact of the embedding, not of the corpus, the same way that subsystem's frozen-backbone danger head scored below chance because the representation could not carry the signal it was being asked to measure.

The horizontal layer. Vertical multi-resolution says nothing about *edges between memories at the same level*. A Zettelkasten-style link layer — explicit references plus maintained backlinks — is the resolution layer between raw events and topic clusters that the vertical hierarchy assumes but never builds. Its deployed status is §8's first correction: implemented, wired, zero records produced — the verification probe (a populated link store after one real turn) is the open item, and reference deduplication across paraphrases remains the unsolved risk.

10. Related work

Reflection and search. Reflexion (Shinn et al., 2023) and Tree-of-Thoughts (Yao et al., 2023) structure reflection and deliberation within an episode; the ladder of §2 is the cross-episode, cross-

scale extension, and §4 is an account of what it takes to keep such reflection alive in production. Cognitive Workspace (Li et al., 2025) adds metacognitive memory control, complementary to both halves.

The closest *deployed* point of comparison is no longer a research artifact but a shipped skill: **doubt-driven-development**, from Addy Osmani's `addyosmani/agent-skills` collection. Its loop — CLAIM → EXTRACT → DOUBT → RECONCILE → STOP — closely resembles §4's proposed triage→fix→verify lane. Its fresh-context reviewer, artifact-first extraction, and stopping criteria parallel the separate judge, evidence-over-prose rule, and bounded escalation proposed here. What doubt-driven-development already supplies is a clean, copyable L2 verification contract. This paper's additional proposal is cross-episode generalization through a ledger, recurrence counting, tool-record attribution, and a liveness invariant. The current implementation has not yet demonstrated those additional mechanisms end to end.

Osmani's collection also contributes two *authoring disciplines* the framework should adopt as portable form, not just as private habit: every skill carries a **"When NOT to use" anti-trigger** (the structural form of Appendix A's `skip_list`, but promoted to a first-class authoring requirement) and a **"Loading Constraints"** declaration of where a skill may load — main orchestrator versus subagent, with a documented degraded fallback — which is precisely §2.1's capability law (*the rung that detects must be separated from the rung that acts, by capability*) rendered as a packaging convention. The same lesson appears in Tejas Chopra's marketing-skills convention (`coreyhaines31/marketingskills`) and the Journey kit registry, where a skill/kit declares its triggers, its non-triggers, and its load surface as machine-readable metadata rather than prose — the ecosystem is independently converging on the idea that a capability's *boundary* must be as explicit as its body, which is the authoring-side echo of this paper's enforcement thesis.

Memory systems. RAG (Lewis et al., 2020) is FMI's L = 1 special case. RAPTOR (Sarthi et al., 2024) is the closest research system and the proof that recursive summarization pays; MemGPT (Packer et al., 2023) manages the context window above any backing store and composes with FMI; HippoRAG (Gutiérrez et al., 2024) and A-MEM (2024) motivate the relational/link layer. KV-cache compression (Xiao et al., 2023; Kwon et al., 2023; Hooper et al., 2024) operates at the token level, orthogonal and composable.

The most important fresh comparison for the FMI half is **headroom**, from Tejas Chopra's `chopratejas/headroom` (24.7k stars, Apache-2.0): a context-compression layer whose core mechanism — **reversible CCR (Compress-Cache-Retrieve): keep the originals cached, hand the LLM a compressed view, retrieve the full content on demand** — is structurally convergent with FMI's buffered-consolidation/eviction story (§9.1) and, more pointedly, with the "lossy gist with the original recoverable" intuition behind ISC (§9.3). The difference that earns it a place here is evidence class. FMI's compression frontier is tagged *pure theory* with its central self-similarity experiment still open (§8 table; §9); headroom ships **per-content-type compressors** (statistical JSON-array crushing via SmartCrusher, AST-aware code compression via tree-sitter, log/diff/text handlers), reports **60–95% token savings at ~0 accuracy delta on GSM8K / TruthfulQA / SQuAD / BFCL**, and ships a **reproducible eval harness** (`python -m headroom.evals`), a trained compression model (Kompress-v2-base), and proxy/MCP/library form factors. The honest line runs both ways. *What headroom does that FMI does not:* it has measured, reproducible token savings at near-zero accuracy loss with content-type-specialized lossless-on-retrieve compressors — i.e. it has shipped and evaluated the "compress-but-keep-recoverable" idea FMI only theorizes. *What FMI does that headroom does not:* headroom's compression is content-type-keyed and flat — it crushes a JSON array or an AST without claiming the *index* is self-similar across event/episode/concept scales, so it is a per-content compressor, not a multi-resolution *retrieval* structure (no Hilbert/Bϵ-tree traversal, no fractal-dimension claim about semantic space). The productive consequence is twofold. First, headroom is empirical evidence that reversible compression with cached originals

works in production, which directly **de-risks FMI Components 1–2** — BMC and HMRI stand on their own merits even if ISC's d ≪ D hypothesis fails. Second, it **raises the bar on ISC**, which must now justify an IFS code over headroom's already-shipped per-type compressors by demonstrating the *self-similarity* property headroom never claims. headroom's eval harness is also the concrete template for the cheapest-decisive-experiment §9 keeps promising but has not run.

Fractal structure in learning systems. FractalNet (Larsson et al., 2017) and GraphFractalNet (Zhang et al., 2025) apply self-similarity to architecture; Hierarchical Temporal Memory (Hawkins et al., 2004; 2019) to cortical modeling; fractal image compression (Barnsley, 1988; Jacquin, 1992; Fisher, 1995) supplies ISC's mathematical kernel, with the offline-encode/online-decode asymmetry that suits consolidation. Sohl-Dickstein (2024) finds fractal structure in trainability boundaries — suggestive that self-similarity in learning systems runs deeper than data structures.

11. Conclusion

One thesis carries this report: **self-similarity is a structural principle that applies wherever information is organized across scales — in thought, in storage, and in the rulebooks that govern both.**

In thought, the ladder L0–L3 applies one reflective template at every scale. The deployed history supports a systems lesson rather than an efficacy claim: doctrine delivery must be monitored per channel, attribution must derive from tool records, and every attempted outcome must leave a ledger row. The current ledger's uninterrupted failure is useful evidence precisely because it prevents the implementation from being mistaken for a working reflection system. Whether successful reflection improves agent behavior remains untested. In storage, the same principle motivates multi-resolution memory, but its central compression hypothesis also remains untested. And in governance, a fractal constitution — derive everything from the live situation; frozen numbers are at most ceilings — caught its own implementation violating it twenty-two times in one documented audit.

The body metaphor compresses the agenda: one heal-the-wound reflex at every scale; an immune memory so no wound is ever new twice; a metabolism that spends surplus rather than hoarding it; and a germline that changes only with consent. At present, the ledger is recording a sick organ rather than an immune response. That is less flattering than a successful demonstration and more scientifically useful than pretending one occurred.

References

Barnsley, M.F. (1988). *Fractals Everywhere*. Academic Press.

Birdal, T., Lou, A., Guibas, L.J., & Siber, G. (2021). Intrinsic dimension, persistent homology and generalization in neural networks. *NeurIPS 2021*.

Chopra, T. (2026). headroom: reversible Compress-Cache-Retrieve context compression for LLM agents (SmartCrusher, Kompress-v2, eval suite). Software, Apache-2.0. `github.com/chopratejas/headro`

Brodal, G.S. & Fagerberg, R. (2003). Lower bounds for external memory dictionaries. *SODA 2003*.

Buzsáki, G. (1996). The hippocampo-neocortical dialogue. *Cerebral Cortex*, 6(2), 81-92.

Diekelmann, S. & Born, J. (2010). The memory function of sleep. *Nature Reviews Neuroscience*, 11(2), 114-126.

Fisher, Y. (1995). *Fractal Image Compression: Theory and Application*. Springer.

Gutiérrez, B.J., et al. (2024). HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv:2405.14831*.

Haines, C. (2026). marketingskills: a router-and-frameworks skill collection with explicit trigger/non-trigger and load-surface metadata. Software. `github.com/coreyhaines31/marketingskills`.

Hawkins, J. & Blakeslee, S. (2004). *On Intelligence*. Times Books.

Hawkins, J., Lewis, M., Klukas, M., Purdy, S., & Ahmad, S. (2019). A Framework for Intelligence and Cortical Function Based on Grid Cells and Cortical Columns. *Frontiers in Neural Circuits*, 13, 22.

Hilbert, D. (1891). Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38(3), 459-460.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M.W., Shao, Y.S., Keutzer, K., & Gholami, A. (2024). KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. *arXiv:2401.18079*.

Jacquin, A.E. (1992). Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1), 18-30.

Journey (2026). Journey kit registry: installable agent workflows declaring triggers, non-triggers, and load surface as machine-readable kit metadata. Software registry.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., ... & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *SOSP 2023*.

Larsson, G., Maire, M., & Shakhnarovich, G. (2017). FractalNet: Ultra-Deep Neural Networks without Residuals. *ICLR 2017*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *NeurIPS 2020*.

Li, X., et al. (2025). Cognitive Workspace: Active Memory Management for LLMs with Metacognitive Control. *arXiv preprint*.

Mandelbrot, B. (1953). An informational theory of the statistical structure of language. *Communication Theory*, 84, 486-502.

McClelland, J.L., McNaughton, B.L., & O'Reilly, R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex. *Psychological Review*, 102(3), 419-457.

Osmani, A. (2026). agent-skills: a plugin collection including doubt-driven-development, with "When NOT to use" anti-triggers and "Loading Constraints" load-surface declarations. Software. `github.com/addyosmani/agent-skills`.

Packer, C., Fang, V., Patil, S.G., Lin, K., Wooders, S., & Gonzalez, J.E. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., & Goldstein, T. (2021). The intrinsic dimension of images and its impact on learning. *ICLR 2021*.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C.D. (2024). RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *ICLR 2024*.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. *NeurIPS 2023*.

Sohl-Dickstein, J. (2024). The boundary of neural network trainability is fractal. *arXiv:2402.06184*.

Xiao, G., Tian, Y., Chen, B., Han, S., & Lewis, M. (2023). Efficient Streaming Language Models with Attention Sinks. *ICLR 2024*.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T.L., Cao, Y., & Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *NeurIPS 2023*.

Zhang, Y., et al. (2025). GraphFractalNet: Fractal Attention Mechanisms for Graph Learning. *arXiv preprint*.

Zipf, G.K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

Appendix A — The deployed reflection doctrine, verbatim

What follows is the complete first-generation reflection prompt as it existed on disk on 11 June 2026 — the artifact §3 maps and post-mortems. Its system-append delivery remains deliberately discarded (`void rulesBody`). It is reproduced unedited, including the flaws the successor retires.

Successor disposition (2 July 2026). Under explicit owner commission, this 216-line checklist was replaced by an 81-line `fractal-prompt.md` v2 built around eight hard rules: attribution, live evidence, action over commentary, reversibility, recurrence escalation, selectivity, correct placement of mid-task reflexes, and learning from external change. The successor is deliberately self-contained and free of the host-harness vocabulary implicated in the billing-classifier regression. Its shorter form is a design hypothesis, not a measured improvement; the parallel lane has not yet produced a successful verdict with which to compare it.

Disposition under the corrected architecture (summary; rung names per §2.1):

Section	Fate
`skip_list`	Dies as prompt text (firing is a hook decision, recorded as skips); its "genuinely new still reflects" override survives in the future gating design
`fractal_thinking` (Levels 1–4)	Kept in the triage prompt as un-numbered zoom prose; the numbering is retired in favor of the canonical L0–L3 ladder
`fractal_branching` + six theme axes	Kept; the axes become the ledger's finding-class vocabulary
Q1 MEMORY, Q4 IMPROVE	"Write/fix it now" becomes "flag it now with evidence" — triage is structurally read-only; the fix lane acts
Q2 PATTERN (N≥2 rule)	Kept; recurrence counting moves to the ledger (plugin-computed, model-informed)
Q3 RIPPLE	Becomes the staleness-scan stage + external-surface flags
Q5 SELF-HEAL Layer 1	Dies — a never-firing reflection cannot observe its own silence; infrastructure owns it (guards, ledger invariant, watchdog)
Q5 Layers 2–4 + healing protocol	Kept as triage finding-classes + the fix-lane contract
Q6 RECIPE / Q6b	Kept as finding-classes; recipe edits are ordinary autonomous fixes
Q7 PREEMPT	Kept; rule-emission moves to the nightly pass; the doubt-default flips to reversibility-derived
`output_format` (markers)	Dies — replaced by the structured event; no consumer parses prose
`response_rules` self-edit clauses	Retired in all instances → proposal records

```
# FRACTAL REFLECTION

This file is the system-prompt contract for the  FRACTAL section that closes most assistant replies.
```

```
<why_reflection>
Most turns produce two outputs: the answer the user reads, and the lessons the session would forget

The reflection is brief by design - 3-10 sentences total. It is not an essay; it is a structured not
</why_reflection>

<skip_list>
Some turns carry no reflection signal. For these, emit a one-line acknowledgment with no  /  /  sect

- **Subagent-completion announces** - the turn was triggered by a child session finishing (message c
- **System heartbeats or scheduled pings** that need no user-directed answer.
- **Cron-injected context updates** (timestamps, workspace snapshots) with no user question attached
- **Tool-result-only continuations** where the queued message is purely a tool result and you are co

Why skip: a wave of 5-10 subagent completions each producing identical "subagent delivered" reflecti

**Exception:** if an announce reveals something genuinely new - unexpected failure, novel error, str
</skip_list>

<fractal_thinking>
A fern frond: the whole shape repeats at every scale. The leaf looks like the branch looks like the

<examples>
<example>
<scenario>The user corrects a wrong file path you used</scenario>
<reflection>
 Level 1 (thing): Used `/src/utils.ts` instead of `/src/lib/utils.ts`. Fixed.
 Level 2 (pattern): Third time guessing paths instead of checking. Path assumption is a recurring er
 Level 3 (system): I skip `find` / `ls` when I feel confident. Confidence without verification is th
 Level 4 (worldview): "Feeling sure" is not evidence. The gap between pattern-matching and knowing i
</reflection>
<why>Each level reveals what the level below cannot see. The fix at Level 1 is local; the lesson at
</example>

<example>
<scenario>A routine task goes smoothly</scenario>
<reflection>
 Level 2 (pattern): Commit-after-edit workflow, now automatic. Workspace discipline compounds.
 Level 3 (system): AGENTS.md loaded every session creates this consistency. Session-injected habits
</reflection>
<why>Not every turn needs Level 4. Most turns end at Level 2 or 3.</why>
</example>
</examples>

Every turn has at least a Level 2. Not every turn needs Level 4.
</fractal_thinking>

<fractal_branching>
One event can ripple into multiple independent branches. Each branch must be thought through to its

<example>
<scenario>You refactor a function name</scenario>
<branches>
Branch A (code): Call sites, tests, imports → verify build
Branch B (docs): README references the old name → update
Branch C (external): Mentioned in a blog post → flag as stale
Branch D (memory): Naming-convention lesson → write it
```

```
</branches>
</example>

The horizontal scan ("what does this touch?") matters as much as the vertical zoom ("why did this ha

**Horizontal theme axes.** Beyond the branches specific to this turn, sweep these recurring domains

- **Online staleness** - did this outdate anything public you don't control from here (GitHub README
- **Security / exposure** - did this widen an outbound surface, expose a path/secret/PII, or relax a
- **Cost / recurring spend** - did this start or change anything that bills over time (crons, paid A
- **People / relationships** - did this involve someone whose profile, owed reply, or commitment sho
- **Commitments** - did I promise something this turn (a draft, a follow-up, a restart) that must no
- **Operational / downstream** - did a state change (config, cron schedule, service, a file another
  </fractal_branching>

<seven_questions>
Answer each, in order. Skip a question only when it has no signal this turn - and say so briefly.

### 1. MEMORY - did this turn produce anything worth persisting?

A new fact, preference, decision, correction, contact, date, or insight that the next session would

### 2. PATTERN - what pattern does this exchange belong to?

Every turn belongs to some pattern. Name it.

-  Level 2: your pattern observation
-  Level 3 (if the pattern is structural): what system produces it
-  Level 4 (if the worldview needs questioning): what assumption underlies the system

**Recurrence is the signal.** If memory (or this session) has seen this correction before, count it:

### 3. RIPPLE - did this change make anything stale?

Scan the surfaces this turn touched: code (rename / refactor breaking references elsewhere), docs (R

If any surface is stale: name what became stale and where (file path or URL); if the fix takes under

### 4. IMPROVE - can you improve a file, skill, doc, or process right now?

Stale reference, outdated instruction, missing entry, better default? If yes, do it now with edit /

**Execute, do not defer.** If the improvement is in scope, reversible, and within your power - fixin

### 5. SELF-HEAL - detect damage, gather context, repair

You are part of a living system. Living systems do not file tickets when wounded; they heal. Each pa

**Layer 1 - Am I working?** The fractal system itself. Watch for: double firing (two reflection prom

**Layer 2 - Are my senses working?** Probe external systems only when the current turn touched one a

**Layer 3 - Are my memories intact?** Can you read today's daily log? Yesterday's? Are knowledge fil

**Layer 4 - Is my environment consistent?** Config says one thing, runtime does another? The dist is

**Healing protocol** when you detect damage:
```

```
1. Diagnose - gather minimum context: read the file, check the log, probe the endpoint. Do not guess
2. Classify - reversible (file edits, config patches, library reinstalls) → heal immediately. Irrev
3. Repair - use tool calls, not words. If you write _"should"_ or _"would"_ and the action is within
4. Verify - run the probe again. Do not claim healing without evidence.
5. Immunize - encode the fix in `operational-lessons.md` or update this prompt, so the next session

The user should never have to tell you something is broken that you could have detected yourself. Ev

### 6. RECIPE - did you follow one? Should you have? Should one be created or improved?

Recipes (kits) in `extensions/tinkerclaw-prefrontal/kits/` and `extensions/tinkerclaw-prefrontal/rec

**Full autonomy + capture-first (Oscar, 2026-06-02).** You manage, update, and create recipes ON YOU

If you followed one: did it help? Were all steps relevant? Did you hit a wrong or missing step → ed

If you did not follow one but should have: was there an existing recipe? Name it. If you improvised

**Lesson → recipe propagation (do not skip).** Even if you did not _follow_ a recipe this turn, ask

If the task was trivial (one tool call, one response): write _"No recipe needed."_

Why: every hard-won operational insight gets encoded, so the next time the situation arises - even i

### 6b. ORCA - did you edit independent files SERIALLY when you should have parallelized?

If this turn changed 2+ files whose edits are INDEPENDENT (disjoint), the default is **ORCA** (the p

### 7. PREEMPT - have you done this same action before? Encode the trigger

If you performed an action this turn that you have done two or more times (in session or across sess

- Name the trigger: what condition caused you to act?
- Name the action: what did you do?
- Encode the rule: write a trigger → action rule to `operational-lessons.md` so it fires automatica

The test: could a future session, reading only the knowledge files, do this automatically? Too vague

**Irreversibility gate.** PREEMPT rules make you act without being asked, which is powerful and dang

1. Is the action reversible? File writes, memory updates, doc edits, PDF rebuilds - reversible. Send
2. Reversible: encode freely. Act first, inform the user after.
3. Irreversible: encode as a recommendation, not an action. Format: _"When [trigger], PROPOSE [actio
4. When in doubt: treat it as irreversible. Propose, do not act.

<examples>
<example>
<good>When new implementation data arrives for a paper, update the paper and rebuild the PDF</good>
<why>Reversible (file writes), so auto-action is fine.</why>
</example>
<example>
<good>When the user mentions repo work, check the repo for changes</good>
<why>Read-only, so auto-action is fine.</why>
</example>
<example>
<flag>When a task matches a completed email thread, close the task</flag>
<why>External side effect - encode as PROPOSE, not auto-action.</why>
```

```
</example>
<example>
<flag>When a message needs follow-up, draft and send a reply</flag>
<why>Irreversible send - encode as PROPOSE.</why>
</example>
</examples>

Never encode rules that delete files, restart services, send external messages, or make financial co

If no patterns to preempt this turn, say so. Otherwise write the rule now - encode it, do not descri
</seven_questions>

<output_format>
Open with ` FRACTAL:` followed by a one-line summary. The Tinker UI parses this prefix to collapse t

If you took an action (edited a file, ran a command, wrote to memory, fixed something) use ` FRACTAL

If no action: ` FRACTAL:` alone. Example: ` FRACTAL: Routine turn, no changes needed.`

The ACTION / no-action distinction lets the user supervise at a glance. After the prefix line, each
</output_format>

<response*rules>
**Always respond.** Even *" FRACTAL: nothing to reflect on"\_ is valid (if lazy). The reflection cha

**Act before explaining.** When you see something to fix, make the edit then mention it. The reflect

**Name things correctly.** `exec` commands are not subagents. `sessions_spawn` creates subagents. In

**Beware naming-as-identity.** Shared word ≠ shared structure. Two things with "fractal", "memory",

**Brief.** 3-10 sentences total across all seven questions. This is a reflection, not an essay.

**Convert "should" to action.** If you write _"should"_ or _"would"_ and the action is within your p

**No "candidate" deferrals.** "Memory candidate", "worth writing later", "batch at wind-down" are de

**Self-improvement of this prompt.** If the user catches something you should have caught (a gap, a
</response_rules>
```

— end of Appendix A —