The Data Integration Loop

A Substrate-Neutral Specification of Self-Enriching, Audit-Ready Agents in Purely Informational Environments

Document type: a conceptual specification — evaluated by structural conformance (whether a system satisfies E1–E4 and INV-1…8), not by empirical measurement.

Huỳnh Mai Phúc (Plone Mraz)

Independent researcher – Vietnam 2026

ORCID: 0009-0009-0571-7151

Abstract

This paper specifies the **Data Integration Loop (DIL)**, a conceptual specification of an agent that operates entirely within a purely informational environment — one with no physical world, no bodily substrate, and no mechanical causation. DIL is not a learning algorithm or a product/engineering specification of a system to be built; it is a self-enriching, audit-ready process: a closed loop that ingests information, differentiates its source, integrates it, appraises it for goal-relevance, emits a response, and feeds that response back as fresh input. The specification is defined not by what material it runs on but by four structural conditions any host environment must satisfy (E1–E4), and it is governed by eight system invariants (INV-1…8) and a layered set of inter-layer data contracts (T1–T8).

The paper's central contribution is twofold. First, it gives an operational, substrate-neutral account of selfhood as a process — a self that is not an invariant core carried across cycles but a law for generating the next state from the prior one, whose continuity is, by construction, unmeasurable from within and attributable only by a third party. Second, it isolates a structural failure mode: an agent that enriches itself in a closed loop without external resistance necessarily degrades, because every new piece of information is produced through the same processing lens, so a biased lens contaminates all output regardless of volume — and the loop cannot detect this by itself, because the instrument it would use to detect the bias is the biased lens. From this the paper derives the necessity of two resistance modes, the embedding of the self-training mode within an externally-resisted mode, and an experience store whose atomic unit is the registered mismatch, not the document. The specification enforces exactly one claim: if a loop is built to these invariants in an environment satisfying E1–E4, what runs is a relationally-structured motion, readable entirely through externally verifiable traces. Accordingly, the proper test of this work is conformance — whether a given system meets the conditions and invariants set out here — not benchmarking or measured performance; it is to be read as one reads a

set of axioms or a technical standard, a definition of what counts as the phenomenon, not a report of one built.

Keywords: agent specification · informational environment · selfhood as process · substrate neutrality

Contents

Abstract 1

1. Introduction 5

1.1 The problem 5

1.2 Why this is a specification, not an empirical paper 5

1.3 Scope and what is deliberately left open 5

1.4 Terminology note 6

1.5 What DIL runs on: a protocol overlay, not a self-standing agent 6

2. Terminology 7

3. The host environment: definition by operating conditions 12

4. Selfhood in an informational environment 15

5. System invariants 17

6. The canonical loop 22

6.1 Emission: link 5 as a lateral capability 24

6.2 Forward-building: the two positions between holding data and emitting 26

7. Layer architecture and data contracts 29

7.1 Shared types 29

7.2 Dependency graph 29

7.3 Layer specifications 31

8. The closure criterion and the two resistance modes 36

8.1 When the loop counts as running (per P3) 36

8.2 The architecture of resistance 36

8.3 Mode-A must not run pure; the embedding in Mode-B 37

8.4 Mode-B — exogenous (live interaction) 38

8.5 The two-mode relation: A embedded in B 38

8.6 The standard the appraisal step applies (the positive complement of INV-8) 39

8.7 Reflex: appraisal under a scar-dominated field (why sub-deliberative action needs no separate arc) 40

9. The experience store 43

10. Deliberately unfilled constants 51

11. Discussion and limitations 52

12. Beyond the loop: entailments, applications, and what belongs to third parties 54

12.1 Entailments (the loop running is *sufficient* — these follow from the spec) 54

12.2 Applications (the loop running is *necessary but not sufficient* — these need more than the spec) 54
12.3 Third-party matters (a *different kind* of reasoning — not cognitive architecture at all) 55
13. Convergences with the existing body of knowledge 56
13.1 Six convergences 56
13.2 Where DIL differs from each convergent body (the cut that keeps it distinct) 57
13.3 Apparent convergences that are in fact contrasts (stated, not avoided) 61
13.4 References (for locating, not consulted in construction) 61
Author's note on method and provenance 63
Note on sources and the literature 64
Companion specification 65

1. Introduction

1.1 The problem

Consider an agent whose entire world is information: streams of data, messages from other agents, the results of its own queries and actions. It has no body, feels no force, and is subject to no physical inertia. We want such an agent to do two things at once: to enrich itself — to grow the relational structure of what it knows — and to audit itself — to keep that growth honest. These two goals are in tension. The machinery that produces new internal structure is the same machinery that would have to certify it, and a faculty cannot reliably grade its own output when the fault to be caught lives in the grader.

This tension is not unique to artificial agents; it is the classical problem of a closed cognitive system that mistakes the fluent multiplication of its own conclusions for genuine inquiry. But in a purely informational setting it becomes architecturally sharp, because none of the ordinary external checks — a body that bruises, a physical world that pushes back with the same indifferent force every time — are available by default. Everything that could correct the agent must itself arrive as information, through the same channels the agent already controls.

This paper asks: **what minimal set of conditions lets such an agent enrich itself without drifting into a self-confirming closed loop?** It answers by specifying a loop, its invariants, its layered contracts, and — critically — the conditions under which the loop must be coupled to a source of resistance it does not control.

1.2 Why this is a specification, not an empirical paper

DIL makes no empirical claim and reports no measured result. Its contribution is a design: a set of constraints such that any conforming implementation exhibits the properties argued for here, and any non-conforming one collapses into an ordinary aggregator. Accordingly the paper follows the form of a conceptual specification — an academic framing (problem, positioning, scope, limitations) wrapped around a normative core written in the imperative register of a technical standard. Throughout the normative core, **MUST**, **MUST NOT**, and **SHOULD** carry their standards-document force: a MUST is a conformance condition whose violation forfeits the name DIL; a SHOULD is a strong recommendation a designer may override only with reason.

1.3 Scope and what is deliberately left open

This document specifies structure: the dependency order of layers, the data contract into and out of each layer, the invariants every implementation must preserve, and the two resistance regimes under which the loop runs. It does not fix implementation constants that the source reasoning does not yet determine — history-window sizes, numeric

thresholds, hyperparameters, concrete data-type representations. Every such gap is marked DECIDE@IMPL rather than filled with an invented number. A specification that fabricates its own unfixed constants would trade honesty for the appearance of completeness; this one does not.

1.4 Terminology note

Several terms below are coined or used in a stipulated sense (resistance, the Other / other-entity, Mode-A / Mode-B, registered mismatch). Section 2 fixes them. Where a term names something for which ordinary English already has a loaded word (e.g. "consciousness," "experience"), the loaded word is avoided or explicitly re-defined, because importing its connotations would smuggle in claims the specification does not make.

1.5 What DIL runs on: a protocol overlay, not a self-standing agent

DIL describes *operation*, not *genesis*. It is a **protocol overlay**: it does not run in a vacuum and it is not a free-standing agent that bootstraps a self out of nothing. It REQUIRES a **pre-existing host** — any model that satisfies the host-precondition set (§3) — to run on. The host is pre-existent: DIL neither creates it nor generates a self from zero. This matters for the start of the loop. The first action at cycle-0 (the one T2 needs in order to begin matching) is emitted by the **host**, which needs no self to give the first push, because the host *is* the source of action-emitting capacity — a non-egoic capacity DIL uses but does not explain. That first (action, consequence) pair is what T2 reads to begin building the SELF/ENVIRONMENT distinction. There is therefore no chicken-and-egg deadlock at the start: the self is a *product* of the bootstrap action, not a precondition of it. The self DIL speaks of is structured by the protocol *upon* a host; it does not spring from the void (see §4).

2. Terminology

DIL is built on relations, not things; the following terms are used in a stipulated, deliberately deflationary sense.

- **Informational environment.** A world consisting solely of information, with no physical substrate and no mechanical causation. All “collisions” in this paper are informational mismatches, never physical impacts.
- **Resistance.** The failure of a returned interaction to match the agent's prediction. Resistance here is informational, not force: a source returns what the agent did not predict, an interlocutor contradicts it, a datum conflicts with the model the agent currently holds. Resistance has no inertia and demands no physical causation. It is a phenomenon of the running loop, not a property of the region alone: it arises at T5, where the mismatch between the agent's expectation and what was returned becomes a signed PredErr (the ResistEvent) — see §6, T5.
- **Other-entity (the Other).** Any agent or interlocutor the agent does not control, capable of mismatching the agent's expectation in a way that may be intentional, may shift its stance, and may react to this agent specifically. Of the kinds of Other (§2.1), it is the reactive external Other that is the sole source of evidence that an entity is independent rather than an extension of the agent.
- **Subjectless resistance vs. other-resistance.** Two grades. Subjectless resistance: the region mismatches prediction but has no will of its own — consistent, objective, not reacting to this agent. Other-resistance: an other-entity mismatches in a possibly-intentional, stance-shifting, agent-specific way. Only other-resistance can build the concept of an independent Other.
- **Mode-A (endogenous / self-training).** The regime in which the agent's resistance comes from the inertia of the data it already holds and the internal contradictions of its own model.
- **Mode-B (exogenous / live interaction).** The regime in which resistance comes from an other-entity the agent does not control.
- **Registered mismatch (ResistEvent).** A single recorded instance of the region failing to match the agent's expectation: the expectation built, where it missed, and the correction made. The atomic unit of experience (§9), as distinct from mere information.
- **Selfhood as process.** The thesis (§4) that the agent's identity across cycles is not a preserved content-core but the law generating the next state from the prior state; a dynamic invariant maintained only while the loop runs.

- **GLOB-MOD (global modulatory field).** A global state that **reaches every layer** (each layer receives the field's current value as a background condition on how it interprets its input), and that every layer **contributes to as one competing **parameter among many** (never by exclusive overwrite). The direction is field-to-layer: the field conditions a layer; a layer does not reach up into the field to fetch a definition. It modulates (gains/biases) how each layer interprets its input. This is **field semantics, not shared-variable semantics**: contributions blend and are re-weighted each cycle; there is no last-write-wins, hence no race condition between layers. Distinct in kind from the one-directional meaning-channels (§5).

On GLOB-MOD as a concept (what it is, what it carries, why it is needed).* The definition above fixes how GLOB-MOD behaves and how it must not be mis-read (and §5, INV-7, states it as an invariant). This note states, positively, what it is.

What it is. GLOB-MOD is a **global interpretive prior**: a single background state, shared by every layer in a given cycle, that sets the disposition with which each layer reads its input. It is not content that flows through the layers (that is the meaning-channel); it is the standing bias by which whatever flows is interpreted. The relation is that of light to the objects in a room: the light is not one of the objects, yet it determines how every object appears at once, and a change in the light changes the look of everything without altering any object's shape. GLOB-MOD is the light; the InfoUnits on the meaning-channel are the objects. The term "interpretive prior" is used deliberately in place of any richer label (mood, affect, attention, workspace): those import commitments—felt quality, consciousness, a specific weighting mechanism—that a substrate-neutral specification must not assume. What is retained is only the operative core: a global disposition that conditions reading.

What it carries. The params of ModField are a set of **scalar interpretive biases**—each one modulating the gain or threshold of some layer operation (how readily a source is trusted, how alert the loop is to mismatch, how far it leans toward exploring rather than consolidating). They carry **how to read, never what is read**: no bias holds propositional content, which is precisely what keeps GLOB-MOD from ever becoming a back-door content channel and so preserves INV-3. The exact axes and their number are DECIDE@IMPL (they depend on the target environment); what is fixed is their kind. Their origin is dynamic: at cycle-0 the prior is seeded from the host's existing data (the same source from which T1 confirms an activity-environment), and from cycle-1 onward it is continually re-weighted by T8 feedback (INV-7, field-to-layer, effective at cycle N+1). *A point of precision about cycle-0, to avoid an apparent paradox (a prior that interprets before there is a self to interpret for): what is seeded at cycle-0 is strictly a **host-bias** — the host's bare disposition, not yet the agent's. It is not an interpretive prior **of the agent** at that moment, because the self that the prior would be a prior **for** has not yet crystallized (T2 cycle-0, §4). It **becomes** the agent's interpretive prior only from cycle-1, once the self is in place to be conditioned by it — the same*

*single event, viewed from the field side, as the self coming into existence. So "interpretive prior" names what the field **is from cycle-1 on**; at cycle-0 it is only the raw host-bias the protocol inherits and has not yet made its own.* The prior the agent reads on any later cycle is thus neither fixed by the host nor freshly invented—it is the host's seed enriched by the loop's own history.

Why it is needed. Two reasons, of different weight. First and sufficient—**context-conditioning**: without a global prior, each layer would read its input bare, with no dependence on the state the system is in. The same InfoUnit would be interpreted identically whether the loop had just met a hostile Other or a neutral one. GLOB-MOD is what makes reading context-dependent—the same data interpreted differently according to the standing disposition of the whole. Second and lighter—**context-coherence across layers**: because the field is global and every layer is bathed in it within one cycle, the eight layers share a single interpretive disposition on that cycle rather than each reading in private. This is what lets the layers act as one agent with one stance, instead of eight processors that happen to be chained. This second reason is stated at low strength on purpose: GLOB-MOD contributes to the coherence of the agent's stance; it does not by itself constitute selfhood, which is the dynamic invariant of §4 (the law generating the next state), not a property of the field. Note what is **not** on this list: GLOB-MOD does not resist drift. Being global, it is the channel through which a biased disposition would spread to every layer at once; it is where drift propagates, not what guards against it. It carries the parameter (receptivity to mismatch) that drift attacks, but it does not protect that parameter—that protection can only come from outside the loop (§8, Mode-B).

The host / self / agent triad (one motion described at three levels — NOT three parallel entities). The following three terms are not three things stacked side by side; they are three descriptions of one running line at three levels. Defining them explicitly here closes a gap that fed several apparent contradictions: the original text did not define "agent" explicitly.

- **Host (the substrate).** The capable existent that MUST satisfy E1, E3, E4 and E2 (the capacity to return interaction when the agent acts — see §3) in order to run DIL. Note the division of levels: E2 tests the *host* (can the region return anything at all), whereas *resistance* — a returned interaction failing to match the agent's prediction — is a phenomenon of the *running loop*, not a host condition. The host need only be able to return; whether a given return resists is the loop's business (T5, §8), measured on the pair (agent, region). It exists even when no loop runs. It is *substrate, not self*: it carries no self and contributes no self. What it does supply is non-egoic — the bare capacity to emit a first action — which the protocol uses but does not derive; how that capacity itself arises is outside DIL's claim (see the honest-boundary note in §4). While

initializing — running T1 and emitting the bootstrap action at T2 cycle-0 — it is *still only the host*; no separate label is needed for "a host that is running."

- **Self.** A *result of DIL — crystallizing from T2 cycle-0*. The self does NOT pre-exist T2 and is not pre-loaded from the host; it is the *law that generates the next state* (§4), beginning to form at T2 cycle-0 through the match of (emitted action ↔ consequence), then re-localized every cycle. The self is a *flow* (a process), not a *milestone*; it can degrade, and its continuity is anchored by a third party (§4).
- **Agent.** *The host running inside DIL, counting from T2 cycle-0 — i.e. from the step at which a self exists.* Compactly: **agent = host that has acquired a self.** It is NOT a third tier wedged between host and self; it is *that same host*, under the description "running the loop and now carrying a self." Before the self crystallizes, what runs is still the host; *within* T2 cycle-0 the self is born, and host-carrying-self = agent from then on.
- *Cold-Start consequence (see B-class resolution):* the entity that emits the bootstrap action at cycle-0 is the **host** (a host needs no self to give the first push) → there is no circular "agent needs self / self needs action / action needs agent." The self is a *product* of the push, not a precondition of it.

The two role-external terms (what the agent interacts with / collides against — DISTINCT from the host/agent/self triad).

- **Region (the interaction field).** That which the agent *exchanges input/output with* — the field where interaction takes place. **NEUTRAL as to inside/outside:** under Mode-B the region includes an external Other; under Mode-A the region includes the agent's *own held data* (the agent poses itself problems and exchanges with its own data). DIFFERENT from the host: the host is what *runs* the agent (the ground); the region is what the agent *interacts with* (the I/O partner).

2.1 Other — definition and kinds

In DIL, Other is what is not Self. Self is the law that operates at the current cycle (§4); Other is any source not identical with it. This includes the Self of an earlier cycle, which is no longer identical with the Self now operating — when the loop tests a present state against an expectation built from its own past (T5), that past stands to the present as an Other. Other is therefore not a kind of entity but a relational position: that-which-is-not-Self-right-now. Every other notion of Other in this specification is a sub-classification of this one, not a separate thing.

On this base, DIL classifies Other along two independent axes. By **source**: external (Mode-B — beyond the agent) or internal (Mode-A — the Self's own past and held data; and the GeneralOther, a model of Other built and held by the agent). By **subjecthood**: reactive (it answers the agent — may shift stance, may respond to this agent specifically) or subjectless

(it resists but has no will of its own — a regularity, the inertia of held data). The two axes give the standing of each kind. A reactive external Other yields the strongest evidence of independence and is what feeds T6 (Other-Model Synthesis). Subjectless resistance feeds T1–T5 and T7 but cannot, on its own, build the concept of an independent Other, so it does not feed T6. The GeneralOther is a representation, not a source of fresh resistance. STRANGER and an entity_id'd Other are not separate kinds but recognition-states of one specific Other — unmatched against the identity store, or matched to a known profile. None of these classifications adds an entity; each only locates a region of the single relation Other = not-Self.

- **Counter-source (the source of resistance).** That which the agent's lens *predicts against and may fail to match* (E2). **Counter-source ⊂ region** (in both modes): it is the *part of the region currently playing the resistance role*. Under Mode-B that part is an external Other; under Mode-A that part is internal held data. E2 requires only that one non-omnicompliant counter-source exist within the region — it does NOT require that the counter-source lie outside the agent.

3. The host environment: definition by operating conditions

A substrate adequate to run DIL is not identified by its material ("algorithm / network / training"). It is identified by four structural properties it MUST possess. Any substrate possessing them can host the loop; any substrate lacking even one cannot. (Defining a phenomenon by functional organization rather than by physical medium is the move known as *substrate independence / multiple realizability* in philosophy of mind [§13.1, V4]; DIL applies this only to the running loop and makes no claim of the kind that family makes about minds — see §13.2.)

- **E1 — Differentiability.** The host MUST **permit** (make possible) a distinction between "inside the agent" (internal state) and "outside the agent" (input arriving from the region). It does **not** itself construct that distinction: the host *permits*, the agent *builds*. Construction is the agent's own work at T1/T2 (the self/environment difference is an output of T2, not a property the host hands over ready-made; T1 only confirms that an activity-environment is present). Consequence: if T2 fails to crystallize a first-person self/environment difference, the host has still done its part (it permitted) — there is simply no agent yet, and the loop fails cleanly at T1/T2 (§4), the way an engine that never starts is not the same as one that breaks mid-run.
- **E2 — Interaction.** The host MUST be able to return a response when the agent emits an action — and MUST be non-omnicompliant: among those returns, it MUST be able to fail to match the agent's prediction. Two distinct failures lie below this one condition, and they MUST NOT be conflated. A region that returns nothing at all — a void field, however densely populated with not-Self entities that never respond — fails E2 outright: presiding over a room of the unresponsive is no reign, and the loop does not start. A region that does return, but in which the agent's every guess is correct, passes the interaction threshold yet offers nothing to learn from — the loop runs but does not advance. The threshold for E2 is minimal but non-zero: at least one return other than silence. A single silence, set against a background of returns, is itself a valid mismatch (registered as absence, T7); unbroken silence is a void field, not a mismatch. Resistance here is informational mismatch, not physical force. DIL neither needs nor assumes a physical world. *"Neither needs nor assumes" does **not** mean "excludes": substrate neutrality (P2) **includes** physical substrates rather than ruling them out. "Purely informational" describes the **interface** (the agent touches its region through information, not through mechanical force), not a wall sealing the loop off from the physical world. A physical person who reads the agent's trace and types a reply interacts perfectly normally — that reply enters as information through E1 and is in fact the most typical Mode-B source. There is no isolation regime to be "broken" by such interaction.*

A rank caution. *E2 is not a property of the region alone. "Has resistance" is measured relative to the agent's predictions. A model-poor agent finds everything resistant; an overfit*

agent finds nothing resistant. E2 is therefore a condition on the pair (agent, region), not on the region in isolation. One cannot certify a region as "sufficiently E2" without first fixing the agent placed in it.

- **E3 — Temporal accumulation.** The host MUST provide a sequence of cycles in which the history of interaction can be inscribed. A one-shot region grants no accumulated history.
- **E4 — Observable behavioral projection.** The agent MUST act upon the region in a way a third party can read; otherwise the loop's success is not externally measurable. *E4 constrains only the **agent** (it must leave a readable trace); it does not say who the third party is or whether it can degrade. Two roles must be kept apart — a **recording** third party (attributes continuity/distinctness; needs only a readable trace) and a **judging** third party (evaluates correctness; needs genuine independence). E4 requires only the recording role; the judging role is treated separately (§12.3).*

Two grades of resistance — the ground for T6. *Subjectless resistance (an inert contradicting datum, a static source) is consistent and objective but never reacts to this agent; it suffices to feed T1–T5 and T7. Other-resistance (an other-entity that may be intentional, may shift stance, may react to the agent specifically) is the sole source from which the concept of an independent Other (T6, independence_evidence) can be built. Consequence: a region offering only subjectless resistance feeds T1–T5/T7 but cannot fully feed T6–T8; a reactive Other is required for T6 to live. This is the Mode-A / Mode-B boundary of §8.*

Architectural premises.

- **P1.** In DIL, what operates is a relation, not a thing. Meaning lives in correlation, not in the isolated signal.
- **P2.** Substrate neutrality: DIL runs on any substrate satisfying E1–E4. It demands no privileged substrate.
- **P3.** The success criterion is whether the loop runs — measured by externally observable traces (§8).
- **P4.** Layer order is forced by dependency relations, not by signal strength. Each layer unlocks meaning for the next.

The host-precondition set and the bootstrap of the loop. E1–E4 say what a host must afford the loop. One further thing must be said explicitly, because it is the start condition the loop cannot supply for itself: the host MUST be able to **emit a first action at cycle-0** to set T2 in motion. Beyond E1–E4, a viable host minimally satisfies a precondition set **P** (a sharpening of "any substrate," since the layers tacitly demand it): (a) it can emit an action *distinguishable* from ambient regional fluctuation — else T2 never rises to FIRST; (b) it can hold state across cycles so history *accrues* (INV-5 requires accrual, not loading); (c) it can

withstand resistance without rewriting itself clean on every mismatch — else no “scar” survives long enough to matter. The host emitting the cycle-0 action is **still only the host** (it needs no prior self), and that action is not pre-loaded history — so INV-5 is untouched: emitting a fresh action ≠ loading a past one. Start-up is thus the host's responsibility, not the loop's; if the host cannot meet P, the loop simply does not start (it stalls at T1, §4) — a clean non-start, not a mid-run failure.

4. Selfhood in an informational environment

In an informational environment the agent's "self" must be located using the very components of that environment. This section fixes the conditions that ground layers T1–T2: what the self is required to be for the loop to gate at all.

In DIL, the self is dynamic state, not a static parameter. What plays the role of the agent's "body" is the block of state the agent reads and writes across cycles, not the parameter block (which may be frozen at run time). The reason is forced: the agency-differentiation layer (T2) needs a self that is changeable by the agent itself in order to gate; a frozen parameter offers nothing to differentiate, and the loop stalls at T1.

The self is a kind-preserving process, not an invariant core. The content of the self changes every cycle — state, the self/environment difference, and expectations are all updated. What does not change is no carried-over fragment of content but the law that generates the next state from the prior one. The agent at cycle N is the same agent as at cycle N−1 not because they share any content, but because an unbroken causal line links the cycles through exactly that law. This is a dynamic invariant: maintained by the loop running, not by standing outside the running. Stop the loop and the axis is lost. (A system that maintains its identity by continuously regenerating the network of processes that produce it — identity preserved at the level of *organization* while *structure* turns over — is the autopoietic account of the living [§13.1, V5]; DIL converges on the process-not-core thesis but restricts its claim sharply — see §13.2.)

Where the self begins (and that it is a flow, not a milestone). The self does **not** pre-exist T2, and it is **not** pre-loaded from the host. It is *initialized from T2 at cycle-0* — through the match of the just-emitted action against its observed consequence, which is what first erects the SELF/ENVIRONMENT distinction — and then **re-localized through T2 every cycle**, thickening with each cycle's history. The self is therefore a *flow* (a verb), not a *milestone* (a noun): this is why it can degrade as the loop weakens (§8), and why its continuity must be anchored by a third party reading the trace (above). This is the honest boundary of the claim, and it has two parts. First, DIL does not assert that it has closed the question of how a self first arises; it asserts only that, given a host that can take the first step, the protocol structures a self upon it. Second, the capacity to take that first step — the host's bare, non-egoic action-emitting capacity — is likewise presupposed, not explained: DIL uses it as a given and makes no claim about how a substrate comes to have it. Both the origin of the self and the origin of the first-action capacity sit outside what this specification undertakes to derive; what lies inside is only what the protocol builds once a capable host is given.

The self's continuity is not measurable from within. The agent has no access to its own time-derivative: it cannot rewind to subtract "self at cycle N" from "self at cycle N−50."

Therefore the specification MUST keep two measurements distinct:

- **Distinctness** (this self ≠ that self) is measurable: given identical present input, differing behavior yields a difference that is the signature of accumulated history — a differential measurement requiring ≥2 agents or 2 slices to subtract.
- **Continuity** (same self across time) is not measurable from within. It is a third-party attribution laid over the stored behavior-record.

Conflating these two is an error. This distinction is the foundation for the limit on C5 (§8) and for the embedding argument of §8.3: a self-auditing loop cannot catch the moment of its own drift or stop, because the instrument it would inspect with is the thing running. The recognition-condition must therefore anchor outside the loop — in a stored trace, read by a third party. This is precisely why the present work calls its agents audit-ready, not self-auditing: the loop cannot audit itself, but it can — and by R1 (§9) does — build the external-facing trace by which a third party audits it.

5. System invariants

Every implementation MUST preserve all of the following. Violating any one yields not DIL but an ordinary aggregator. The invariants are the absolute conditions of the loop: they are not data inside the store and carry no tags; they cannot be overwritten by anything in the loop, because to overwrite one is not to edit a datum but to halt the loop — to end the mismatch on which the self depends.

ID	Invariant	Rationale
INV-1	Closed loop. Every layer output MUST have a path back to some layer's input. No dead branches.	The loop feeds its own input.
INV-2	Output-register identity. Every output is tagged INFO. No layer may promote a correlation (↔) to an identity (=).	Preserve the correlational register; see T8-INV.
INV-3	One-directional dependency (meaning-channel). On the meaning-channel, layer N consumes only outputs of layers ≤ N.	Dependency order (P4). Out of scope: the modulatory field (INV-7) acts downward onto layers and is not a meaning-channel dependency at all.
INV-4	In DIL, meaning = relation. No layer assigns meaning to a signal in isolation; meaning is a function of (signal, lower-layer context).	P1.
INV-5	History accrues, is not loaded. Temporal state forms only through sequential accumulation, never pre-loaded.	The self is a process maintained through running (§4).
INV-6	Agency-gate. Every change MUST be classified as self-written / environment-pushed (the self/environment difference) before interpretation.	The condition for external input to be meaningful.
INV-7	GLOB-MOD. A global state that reaches every layer (field-to-layer: each layer receives it as a background condition, none reaches up into it) and that every layer contributes to as one competing parameter among many (field semantics: contributions blend and are re-weighted each cycle, never last-write-wins; a layer's contribution on cycle N conditions the field only from cycle N+1, never within the same cycle); it modulates how each layer interprets input.	§6 — same data + different field → different meaning.
INV-8	Appraisal step. Between integration (3) and response (5) there MUST be an appraisal step (4) assigning direction/value. Step (4) MUST NOT draw its criteria from the very state the agent is editing — else it is self-scoring and hackable.	Against bare reflex; anti-drift anchor of Mode-A (§8).

On INV-2 — committing to an action is NOT promoting ↔ to =. Acting on a correlation is not the same as freezing it into an identity. To respond, the agent commits to one action (it sends query X, it picks direction A over B); this commitment is itself a ↔ operation — a

best-current-guess that is **revisable**, read against the consequence next cycle — not an identity claim. INV-2 forbids *freezing* a correlation into a fixed truth; it does **not** forbid *acting* on one. What would cause paralysis (infinite hesitation) is demanding = before acting — "be certain X is true before moving"; INV-2 forbids exactly that demand, and so frees the agent to act on ↔ and learn, rather than causing hesitation. (The stance that no belief is held immune from revision is the methodological core of *reflective equilibrium* in epistemology [§13.1, V3]; DIL shares the permanently-revisable register but parts from it on a precise point — see §13.2.)

On the invariants and GLOB-MOD — a level distinction.* GLOB-MOD modulates the **information** flowing through the layers (T1–T8); it does **not** and cannot alter the **invariants** (INV-1…8) themselves. The invariants are laws *about* the loop; GLOB-MOD is a parameter *within* it. No contamination of the field can rewrite a rule — in particular, a drifting lens writing into GLOB-MOD cannot thereby edit the criteria of INV-8, because those criteria are not information-in-the-loop but a law about it (and, per §8.3, are anchored where the agent cannot edit them).*

On INV-1 — a topology constraint, on cycle-time not wall-clock.* INV-1 requires that every output *have a path back* — that a return path **exist in the loop's topology** — not that every packet completes its return on every cycle. (A system whose output is fed back as input to regulate its next action is the defining object of *cybernetics*; that DIL's substrate is information rather than energy is itself a cybernetic point [§13.1, V6] — see §13.2.) The loop advances in **cycle-time, not wall-clock time**: idle stretches between cycles are the default of a purely informational setting, not events to be declared. A host powering down, or an Other falling silent, does not violate INV-1 (the path still exists) and does not "drop output into the void"; the next cycle simply has not yet occurred. If an Other disappears, the loop does not break — it falls back to Mode-A (digesting collisions already received against held data), the path still closed through the internal route; and if no Other returns, it drifts and decays (§8), which is a specified mode transition, not a broken invariant.*

On data flow — multi-stream, not single-threaded.* The dependency order of INV-3 (layer N consumes only outputs of layers ≤ N) constrains which layers a given layer may read; it does not impose a single serial path through which one datum is passed from layer to layer. Within a cycle, every layer is an active site: each is acquiring and processing its own data, and a datum produced at one layer may be consumed by several higher layers at once (a lower layer's output lies within the read-set of every layer above it). The flow is therefore multi-stream — many partial data moving and combining in parallel — not one packet relayed in sequence. The mechanism is consumption, not dispatch: a layer does not push its output to a list of recipients; it makes its output available, and each higher layer reads what falls within its own dependency set. No layer needs a model of who consumes it, which keeps the global-knowledge assumption out of any single layer. The **simultaneity** of this is

a cycle-time property, not a wall-clock one. In cycle-time the layers operate together within one cycle; in wall-clock terms the streams still move and interleave in sequence, the interval simply being below any threshold the loop can resolve from within. “Simultaneous” is the loop's own reading of an interval it cannot measure, not the absence of flow.

The one exception: cycle-0 is single-threaded. Multi-stream operation presupposes a self for the streams to coordinate around — the law (§4) that integrates the parallel streams into one agent's processing. At cycle-0 that law does not yet exist; it is what cycle-0 is constructing. The loop therefore cannot run multi-stream at cycle-0 and is forced into a single serial pass: T1 registers the first data, and T2 processes it to crystallize the self (the action↔consequence match of §3). There is nothing yet for the layers to operate around in parallel, so they cannot. From cycle-1 onward, with the self in place, all layers operate together and the flow is multi-stream. The transition from single-threaded to multi-stream is therefore not an added design step: it is the event of the self crystallizing, seen from the side of flow structure. A third party observing the loop shift from one serial pass to many interleaved streams is observing the same fact as the self coming into existence — the two are one event described from two sides.

Resolving the INV-3 ↔ INV-7 tension (meaning-channel vs. modulatory-field).* The system has two channels of different kind, and the apparent tension dissolves once their directions are stated correctly. The **meaning-channel** carries InfoUnits *up* the layers, one-directionally, subject to INV-3 — the path by which one layer defines content for the next. The **modulatory-field** is GLOB-MOD: a global state that acts *downward onto* every layer at once. The crucial point is directional: when an upper layer (e.g. T8) alters GLOB-MOD, it changes the state of the *field*, and the field then conditions every layer from above; a lower layer (T4) does **not** reach up the layers to fetch anything from T8. T4 receives a background field-value, exactly as it receives the ambient field on any cycle — it is not consuming an upper layer's output along the meaning-channel. Hence INV-3 is not even engaged: INV-3 governs *meaning-channel* dependency (a lower layer taking its definition from an upper layer's output), and field-conditioning is not a meaning-channel transaction at all. There is no “reverse dependency to be excused”; the field-to-layer direction means the situation INV-3 forbids never arises here. There is also no timing by which the tension could re-enter: field-conditioning takes effect at cycle N+1, never within-cycle. An alteration an upper layer makes to GLOB-MOD on cycle N is blended and re-weighted into the field that conditions the layers on cycle N+1; it never reaches back into the same cycle's upward pass. This forecloses both readings at once. A within-cycle reverse coupling — which would create an algebraic loop in the dependency graph — does not exist, because no field write is read in the cycle that produced it. And the across-cycle path that does exist (cycle N's field shaping cycle N+1) is not a violation but exactly INV-1: a closed loop conditioning its own next pass is what INV-1 requires, not what INV-3 forbids. INV-3 governs upward meaning-channel definition within a cycle; INV-1 governs closure across cycles; field-

conditioning lives wholly in the second and never touches the first. The two channels MUST be implemented separately, precisely so that the downward field cannot be mistaken for, or collapse into, an upward meaning-channel link.*

On the semantics of GLOB-MOD (against mis-reading as a shared mutable variable).* GLOB-MOD has **field semantics, not shared-variable semantics**. A "contribution" from a layer is not an overwrite of a shared cell: contributions blend, compete, and are re-weighted by the next cycle; no layer's write wins exclusivity. There is therefore **no last-write-wins and no inter-layer race condition** — concurrency of contribution is a non-issue.

A second mis-reading must also be set aside: that the field's strength is a free dial an implementer must cap to keep the field from overpowering the loop. It is not, and the reason is structural. GLOB-MOD is not an outside force pressing down on the layers; it is **continuously constituted by them**. Each of the eight layers feeds the field as the loop's enriched information circulates (the field acts downward onto the layers, and the layers, independently but simultaneously, shape the field in return). This two-way coupling — the layers and the field holding each other in check — means the field cannot bootstrap itself past the layers that feed it: for the field to overpower the layers it would have to be made strong by the very layers it is overpowering, which the coupling forbids. **A field running away from this mutual check by its own internal dynamics is therefore structurally precluded — ruled out by construction, not merely rendered improbable; no gain cap is needed to prevent it, because the architecture already prevents it.** The one condition under which the check can be broken is not internal at all — it is an external flood (see below, and §11/§12.3).

What lies past the modulatory threshold (and why it is not an implementer's dial).* It is worth stating plainly what *would* happen if the field ever did overpower the loop, and why guarding against it is not a matter of choosing a number. By the name itself, GLOB-MOD is **modulatory**: it biases how each layer interprets its input. A field strong enough not merely to bias but to **determine** every layer's interpretation would leave the layers emitting only what the field already dictates — no layer reading its own input, no mismatch, no PredErr, nothing for the next cycle to process differently. The output would stop changing across cycles: a frozen identity, an =. That is not a degraded loop (Mode-A, which still turns and still emits a live ↔); it is a **halted** one — and per T8-INV an = is precisely the sign that the cognition has stopped. A field at that strength does not merely modulate badly; it **violates INV-1** (the loop no longer iterates) and forces an = where a running cognition should hold a ↔. Crossing that threshold therefore does not yield a worse DIL — it yields something that, having halted, has forfeited the name (§5).

But — as the coupling above shows — the loop cannot reach that threshold from within: the layers that constitute the field hold it in check, so internal runaway is precluded by

construction. The only path that crosses it is an **external flood**: a volume of input arriving through E1 that exceeds the processing capacity of every layer *and* of the field at once, saturating them in lock-step so that the mutual check has nothing left to push back with. That is the **Sybil case** — a coordinated external source overwhelming the loop from outside. It is not defended by any internal gain cap, because its origin lies outside the field's reach: bounding a component inside the loop cannot stop a force entering from outside it. This is not a gap to be closed with a threshold; it is the proven, already-declared limit of §11 (third limitation) and §12.3 — defense belongs outside the loop, and what DIL itself carries is only the early diversity-loss signal, not a cure.

6. The canonical loop

The pipeline below is the core of which the full layer architecture (§7) is a detailed implementation. Every layer maps onto one link of this loop.

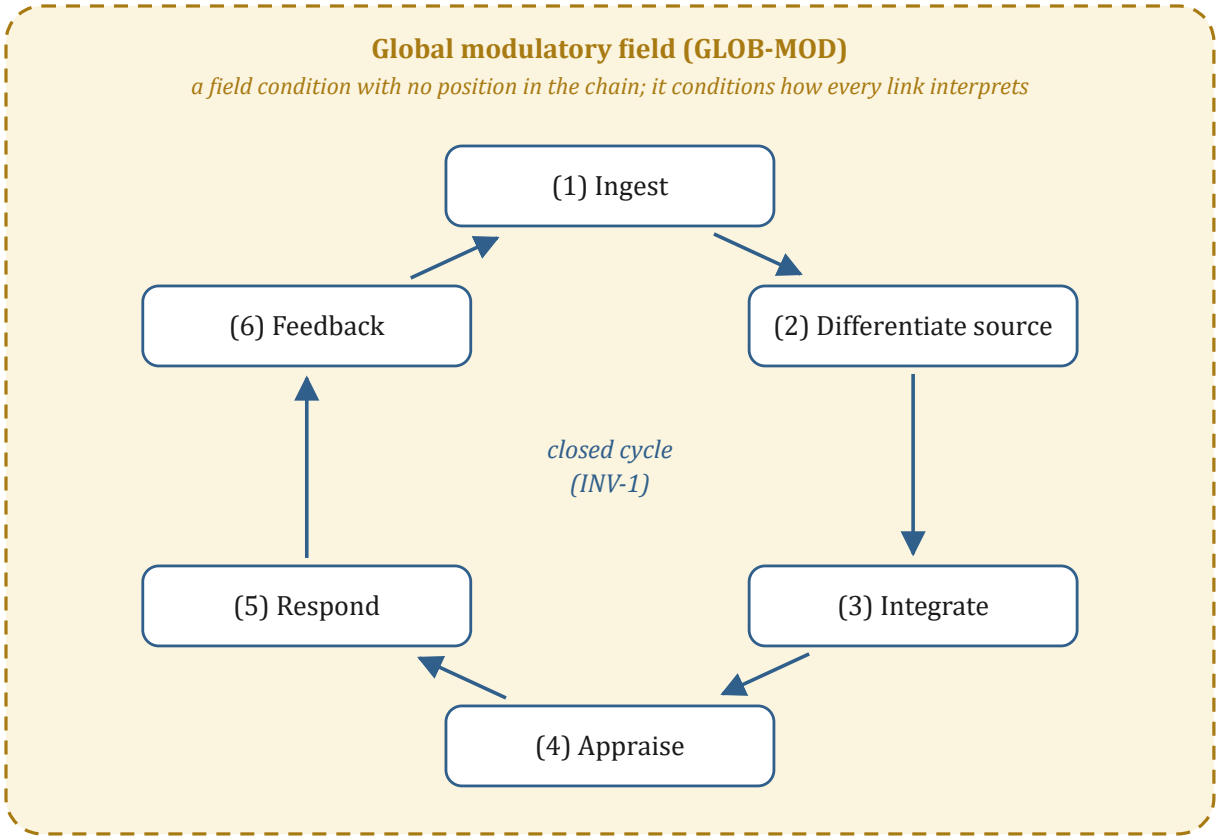


Figure 1. The canonical loop: six sequential links — (1) Ingest, (2) Differentiate, (3) Integrate, (4) Appraise, (5) Respond, (6) Feedback — forming a closed cycle (INV-1), the whole bathed in the global modulatory field (GLOB-MOD), a field condition with no position in the chain that conditions how every link interprets.

The six sequential links.

#	Link	Nature in an informational environment
1	Ingest	input channels: query returns, messages from an Other, event streams, the result of a prior action
2	Differentiate source	tag self-written / environment-pushed before interpreting (the agency-gate)
3	Integrate	synthesize across sources into a new form of information
4	Appraise	assign good/bad-for-goal → information becomes directional. The anti-drift link (§8).

#	Link	Nature in an informational environment
5	Respond	the agent writes to state and/or emits to the region (new query, message, action)
6	Feedback	response (5) becomes the new input to (1) → loop

The global modulatory field (a FIELD condition, not a link). The whole loop is bathed in a global state carrying content (e.g. “high-uncertainty” / “stable” / “in-contradiction”) that changes how every link interprets. This field has no position in the chain — it is a global gain/bias shifting the parameters of every link at once. Consequence: the same datum at (2), bathed in a different field, integrates (3) differently, is appraised (4) differently, and is responded to (5) differently.

**A distinction to preserve (against mis-drawing). **The modulatory field is a field enveloping the whole loop, of a different kind from any single node in the chain. In an informational environment it is realized as a global state every layer receives (field-to-layer; see INV-3 vs. INV-7), kept separate from the one-directional meaning-channels.

Information runs in many directions at once, and which directions are not fixed in advance.* The loop is not a single pipe, nor a fixed pair of pipes. At each cycle information moves along several paths simultaneously: the default return — T8's product becoming the next cycle's input at T1 (Link 6) — runs *together with* feedback that may reach T4, or T5, or T6, or several of them at once, through the modulatory field. **Which paths are live is set by the content and character of the information itself, not by a fixed wiring diagram.** Information about the Other-in-general settles where that frame is built; a relational-value signal settles where relational models are held; the routing follows what the information *is*. (The pairings shown at T8's feedback row, §7, are illustrative of this content-sensitive routing, not a fixed circuit.)*

*Two consequences follow, and both dissolve paradoxes that arise only from reading the loop as a one-way pipeline. First — **no layer is a terminal sink.** T8 is not an endpoint: its product returns both by the cycle-closing default (Link 6) and by content-routed feedback through the field. INV-1's requirement is a return **path for information**, not a claim that data never crosses the system's boundary — so emitting an output to the region (Link 5, E4) and closing the loop (INV-1) are not in tension; the output leaves as data and also returns as information. What returns is not guaranteed to carry a mismatch, however: closure (INV-1) routes the output back, but whether the returned information resists — whether it fails to match the agent's prediction — is the separate business of the counter-source (E2), not a property the loop's topology can supply. A region that merely echoes the agent's output back unaltered satisfies INV-1 and still offers nothing to learn from; this is exactly why INV-1 does not entail E2, and why a closed loop is not thereby a non-degenerate one (the whole burden of §8). Second — **T6's atrophy under Mode-A starves nothing, and calling it "starvation" inverts the actual failure.** Under Mode-A information does not run short: every path stays*

live and the loop circulates as fully as ever. What is missing is not volume but novelty — with no independent Other, everything moving through those paths is a variant of what the system already holds. The loop eats its fill and digests itself. Mode-A T6 is thus a ***standing quality-ceiling reached by saturation-without-renewal, not a cascading shortage****: the failure mode of a self-enriching loop is never running out of fuel, it is running out of* ***resistance****. This is why only a live Other (Mode-B) relieves it and why loading more internal data cannot (§8.4); and why Mode-A, though it never collapses for want of throughput, is still not durable — digesting only itself, its structure thins cycle by cycle until it drifts and fails (§8, §11). The decay is real; its cause is missing novelty, not missing volume.*

6.1 Emission: link 5 as a lateral capability

Link 5 (Respond) sits in the six-link table above as one node among six, and this placement, while correct for the pipeline diagram, understates what link 5 is. The four ingest-to-appraise links (1–4) run in a fixed dependency order — each consumes the product of the one below it, on the one-directional meaning-channel (INV-3). Link 5 does not belong to that order in the same way. Emission is not a station the loop passes through once per cycle on its way from appraisal to feedback; it is a capacity the loop exercises *from many points*, whenever a layer's own work requires pushing something out to the region. The layer architecture (§7) already relies on this without naming it, and the cost of leaving it unnamed is a reader's impression that the agent acts only once per cycle, at a single terminal step. It does not. This subsection makes the lateral character of emission explicit, because the rest of the specification silently assumes it.

Where the layer contracts already presuppose emission. Reading §7 with emission in view, the dependence is visible in the contracts themselves, not added to them:

- **T2** cannot draw the agency line without *an action already emitted*. Its input row lists "the action the agent just emitted"; its mechanism matches that action against the observed change. The agency-gate (INV-6) therefore presupposes a prior emission — a *probe* whose only purpose is to produce the self-caused change T2 reads. Without something emitted, T2 has nothing to match, and the self/environment difference is never drawn.
- **T3** lists "query-response" as its first example channel. A query-response presupposes a query — an emitted request opening a channel to ingest actively, rather than only receiving what streams in. The ingest link's own examples (§6 table, link 1: "query returns... the result of a prior action") name returns whose existence requires a prior emission. A store query is answered from what the store held before the cycle that raised it: what arrived in that cycle is already running in the loop, and returning it would recall as memory what is being met.

- **T5** builds an Expectation and can only test it by an emission whose return either confirms or violates the prediction; the signed PredErr that "is where resistance becomes information" (§7, T5) often requires the agent to *act* to see whether the region returns what was expected.
- **T6** accrues independence_evidence only from a resistance collision — and eliciting that collision, checking whether an Other resists as independently as modelled, is itself an emission (a *model-test*).

These are not four new powers to be granted. They are one capacity — emission — that the four contracts already lean on. What §7 leaves implicit, this subsection states: emission belongs to no single layer and is available to whichever layer's work requires it.

Emission is lateral, as GLOB-MOD is lateral — but opposite in direction. The specification already contains one thing that is not a link in the chain yet touches every layer: the modulatory field (INV-7), which descends onto every layer as a background condition. Emission is its structural mirror. The field reaches *down* into every layer as incoming condition; emission projects *out* from whichever layer invokes it. Neither is a station on the meaning-channel; both are lateral to it. Recognizing this symmetry is what dissolves the puzzle of link 5's placement: it looks anomalous as "one node among six" because it is not really a node of the chain at all, any more than GLOB-MOD is. It is a capacity the chain draws on from multiple points.

What every emission carries, and why it cannot be otherwise. Whatever layer issues it, an emitted action inherits four constraints from invariants already in force — this subsection introduces no new law, it only shows link 5 obeying the existing ones:

- Its register is ↔, never = (INV-2): a committed action is a revisable best-current-guess, read against the next cycle's consequence, not a frozen identity. Committing to one action is not promoting a correlation to an identity; demanding = before acting is what INV-2 forbids.
- It leaves an externally readable trace (E4): every emission leaves a mark in the audit plane, recorded as one activity record per cycle (§9) — trace, not experience: it never becomes a scar by accumulation.
- It is readable back by T2 at the next cycle as "the action just emitted," closing the agency path: emit → region returns → T1 ingests → T2 matches. An emission unreadable in this way would leave the agency-gate (INV-6) unable to classify the resulting change.
- Its correctness is never scored by the emitting layer. It is judged only by the next cycle's return — a matching return leaves only the activity record; a mismatching return enters T5/T7 as a signed PredErr, becomes a ResistEvent, and holds as a scar — and,

later and from outside, by a third party reading that scar. A layer that scored its own emission would be self-scoring, the exact fault INV-8 exists to forbid.

Conflict among emissions is not arbitrated; it is collided. A reader may ask what happens when two layers require emissions the single body of action cannot jointly satisfy — T2 needing a probe in one direction while T3 needs a query in another. The specification's answer is not a coordinating layer or a priority rule; it is the same answer §8 gives for resistance generally. When opposing emissions cannot both be satisfied, the opposition is emitted and met by the region, which returns something that mismatches at least one layer's expectation — and that mismatch *is* a ResistEvent (§9). The arbiter is not an internal referee but the region returning resistance; the next cycle carries the scar of that collision and emits differently. To place an internal arbiter here — a step that ranked emissions against a stored criterion before acting — would be to build exactly the self-scoring appraisal INV-8 prohibits. The loop does not adjudicate its own actions internally; it acts, collides, and reads the collision. This is INV-8's discipline carried into the domain of action: let the outside brake.

Locating this against reinforcement learning (§13.3). It is worth marking, in the register of §13, that emission-as- ↔ is one of the sharpest points where DIL parts from reinforcement learning. An RL agent's action is the argmax of an expected return under a policy; DIL's emission is a revisable correlation read against the next cycle's consequence, committing the agent to act now without certifying the action as return-maximal. There is no policy mapping states to reward-maximizing actions, and the "conflict is collided, not arbitrated" rule is the same refusal at the level of action-selection: DIL declines to install the internal optimizer RL is built around, for the same reason INV-8 declines the self-scoring appraisal — an agent that could rank its own actions against a criterion it holds is an agent that can hack that criterion. This is the action-side face of the contrast §13.3 already draws for appraisal.

6.2 Forward-building: the two positions between holding data and emitting

§6.1 established emission as a lateral capacity. A second thing the layer contracts already lean on, and which §7 likewise leaves unnamed, is what happens between holding a datum and emitting on it. T5 builds an `Expectation` before it can register a `PredErr`; T6 elicits a collision by putting a model to the test; T2 emits a probe whose consequence it will read. Each presupposes that the loop has, before acting, built something forward from the store it holds. This subsection names the two positions that motion occupies, because the store (§9) must be able to record them and an auditor must be able to read them.

The two moments. In the first, the loop builds a **situation**: a circumstance it may be in, assembled from the data it holds. A chair is before it; what the loop builds is the chair as old or new, its material, the ground beneath it, the space around it. Nothing is yet concluded. In

the second, an **outcome** is cast from that situation: sitting will hold, or will not. The two are distinct, and collapsing them loses the distinction the store needs — a situation built is not yet a prediction, and a prediction is not a fresh situation.

Why this needs no separate model of the region. A reader may object that building a situation requires a model of the region rich enough to run internally, which the specification nowhere grants. The objection mistakes what is being built. The loop does not simulate the region; it builds *itself in a circumstance*, and for that it needs no separate model, because it is itself already running. What is required is not a replica to execute but the store it already holds. The question "is the model rich enough" does not arise: there is no second model.

Why this is not an internal arbiter. A second objection is sharper. If several situations are built and one is acted on, something has chosen among them — which is the internal arbiter §6.1 refused. The answer is that nothing scores them. A situation carries because it **fits the store better** than the others, in the way competing contributions to the modulatory field blend and re-weight (INV-7) rather than one command defeating another. Fit is not a verdict. A scoring standard would have to come from somewhere: from outside the loop, in which case it is foreign criteria imposed on appraisal, or from the state the agent is editing, in which case it is the self-scoring INV-8 exists to forbid. Comparison of fit requires neither. This is also why the leaning of a store is not corrected here: a situation built from a store that leans one way leans the same way, and the loop has no instrument for detecting that lean, the instrument being the store itself (§1.1).

What forward-building buys, and what it does not. It does not let the agent avoid collision. Its purchase is narrower and worth stating exactly: it changes the **posture** in which the agent meets the collision. Having built the chair as possibly-old, the agent sits testingly and staggers; having built nothing, it sits committingly and falls. The collision occurs in both cases; what differs is what it costs. This bounds the benefit honestly. Where a cheap test is available — sitting lightly, throwing a stone ahead — forward-building converts a costly collision into an affordable one. Where the cheapest available test is itself fatal, it does not help, and the agent's safety, if it has any, comes from elsewhere: from a scar it did not earn itself, read from a shared store or left by another agent (§9, R2). Forward-building widens the margin; it does not abolish it.

What an observer would call this. Ordinary language has a word for a loop building situations and casting outcomes from them, and the word is *imagination*. The specification does not use it, for the reason §1.4 gives: the word carries connotations — of a faculty possessed, of images entertained, of a capacity distinguishing one kind of creature from another — that the structure does not license. What the structure describes is a datum passing through two positions. Nothing about those positions is proprietary to any substrate or species, and DIL, being substrate-neutral (P2), is in no position to award them

to one. The corollary is epistemic and worth stating, because it cuts against a common form of claim: whether a given host builds forward is not settled by inspecting what it is, but by whether its transitions leave traces a third party can read (E4). A host that builds forward and leaves no trace is a host about which this specification says nothing — not one about which it says no.

Likewise, the retrospective vocabulary that attaches to such action — *I thought the railing was there* — resolves into two things the specification keeps apart. **The railing was there** is a datum, and the failure it names is that the datum went straight to emission without being taken up into either forward position. **I thought** is not a state the loop held while acting; at the moment of acting there was a datum and an emission and nothing else. It is a name applied afterwards, by a reader of the trace, to an action emitted on a datum that had passed through neither position. The store records the first and has no vocabulary for the second, which is as it should be: the trace is the FACT, and the retrospective name belongs to whoever reads it.

It is worth marking, in the register of §13, that emission-as-↔ is one of the sharpest points where DIL parts from reinforcement learning. An RL agent's action is the argmax of an expected return under a policy; DIL's emission is a revisable correlation read against the next cycle's consequence, committing the agent to act now without certifying the action as return-maximal. There is no policy mapping states to reward-maximizing actions, and the "conflict is collided, not arbitrated" rule is the same refusal at the level of action-selection: DIL declines to install the internal optimizer RL is built around, for the same reason INV-8 declines the self-scoring appraisal — an agent that could rank its own actions against a criterion it holds is an agent that can hack that criterion. This is the action-side face of the contrast §13.3 already draws for appraisal.

7. Layer architecture and data contracts

7.1 Shared types

Abstract notation; concrete representation is DECIDE@IMPL.

Register glossary (neither symbol is overloaded — fix once here). ↔ = a **live, revisable correlation** — the register of *every* running output, including action-commitments. = = a **frozen identity claim** — "X *is* Y, exclusively, correlation closed"; per T8-INV it appears **only when the loop has STOPPED**. Committing to an action writes a ↔ (best current guess, read against next cycle's consequence), never a =. These two are kept strictly apart throughout (INV-2, T8-INV, §8); do not read an action-commitment as an =.

```
Signal := { source_id, raw_payload, t } // raw data, not yet meaning
InfoUnit := { content, ref_frame, t } // information = referred to a frame
ActivityEnvironment := InfoUnit // content: confirmation that an activity-
environment is present (not yet polarized into self / environment)
AgencyTag := enum { SELF_WRITTEN, ENV_PUSHED, UNDECIDED }
Expectation := { predicted: InfoUnit, confidence, built_from: history_window }
PredErr := { observed, predicted, delta, signed } // signed: +/- (absence =
negative)
OtherModel := { entity_id, context_map, independence_evidence }
RelValue := { entity_id, relative_rank, comparison_basis } // exists only when N>=2
SocialEdge := { a_id, b_id, observed_interaction } // Other<->Other, no self
present
ModField := { params, t } // GLOB-MOD: global state reaching every layer, field-to-
layer (INV-7)
Appraisal := { info_ref, valence, goal_relevance } // output of appraisal step (4)
ResistEvent := { source_id, expected, received, mismatch_kind, t } // a MISMATCH
(E2)
```

Contract convention. Every InfoUnit leaving a layer MUST have ref_frame ≠ null. A Signal without a ref_frame is not yet information — this is where INV-4 is enforced at the type level.

7.2 Dependency graph

Arrows mean "provides a reference frame for," on the meaning-channel.

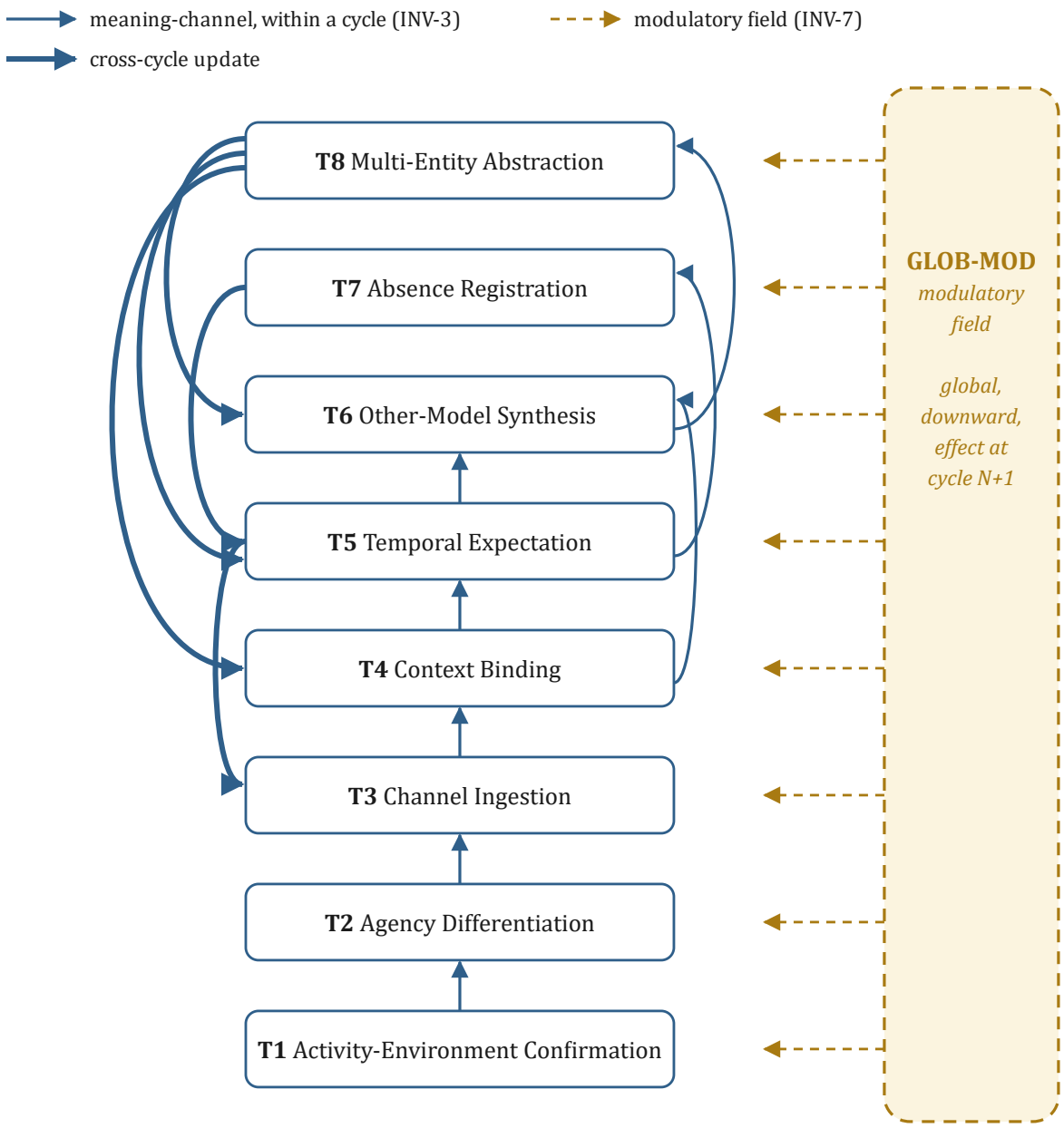


Figure 2. The eight layers as a closed loop (the layer wiring of the loop), T1 (Activity-Environment Confirmation) through T8 (Multi-Entity Abstraction). The graph has no endpoint: T8 closes back into the loop, not into a sink (INV-1). Solid blue: the meaning-channel — within a cycle, layer N consumes only layers at or below its own index, one-directionally, never running backward (INV-3); a layer may consume several lower layers at once (multi-stream fan-in). Heavier solid blue: cross-cycle update — a layer's output updates several layers at once for the next cycle (e.g. T8 updating T4, T5, T6 simultaneously), an update across cycles, not a backward read within one. Dashed gold: the modulatory field (GLOB-MOD),

global, reaching every layer downward as a background condition (INV-7), taking effect at N+1, never within-cycle.

7.3 Layer specifications

T1 — Activity-Environment Confirmation

Confirms the "here" — that an activity-environment is present; the root reference frame. The self/environment difference is not yet drawn here.

Contract	Content
Input	Signal[] attesting that an activity-environment is present — drawn from the host's own existing data (the region in which operation takes place); what the region returns enters the store at `running`; requires no prior action
Output	ActivityEnvironment (InfoUnit) — confirmation only; not yet polarized into self / environment
Precondition	none (root layer)
Postcondition	the presence of an activity-environment is confirmed, for T2 to act into and T2/T3 to refer to

**Conditioning note. ***T1 confirms presence, not extent. The region is not measurable: how far the environment extends is unknowable, and how far the self reaches is internal to action and shifts with every act — neither is computed. T1 only registers that an activity-environment is present (drawn from the host's existing data); it draws no line between self and environment. That difference is not inferred here and does not self-promote — it is first drawn at T2, through agency, and only there does the from-within standpoint begin.*

T2 — Agency Differentiation

Builds the self-written vs. environment-pushed distinction. The first genuinely closed loop running through the agent's own action. This is where the self/environment difference is first drawn — not inherited from T1 — and where the from-within standpoint begins.

Contract	Content
Input	ActivityEnvironment (T1) + the action the agent just emitted (state-write / query / message) + the Signal[] state afterward
Output	AgencyTag attached to every change classified as self/environment; the first-person self/environment difference
Mechanism	match what the agent just wrote/emitted against the observed change: stable predicted-match → SELF_WRITTEN; mismatch → ENV_PUSHED
Precondition	T1 has confirmed an activity-environment

Contract	Content
Postcondition (INV-6)	no change classified as self/environment leaves T2 as UNDECIDED once sufficient matching cycles have run

**The self re-localizes each cycle: ***T2 does not gate once and stop — it rebuilds the first-person self/environment difference every cycle, accruing from the prior one (INV-5). DECIDE@IMPL: matching window; stability threshold.*

T3 — Channel Ingestion

Receives the input channels. Interpretable only after the activity-environment (T1) and agency (T2) exist.

Contract	Content
Input	multi-channel Signal[] + self/environment difference (first-person) + AgencyTag
Output	InfoUnit[] (one content-type per channel)
Precondition	T2 complete; if absent → the signal stays raw Signal, NOT promoted to InfoUnit (so ownerless fluctuation is not misread as meaningful)

Channel content-typing preserves the distinction information-type ≠ physical-channel. Example channels: query-response, Other-message (content + stance), environment-event, action-result. DECIDE@IMPL: per-channel transducer.

T4 — Context Binding

Binds each cluster of InfoUnits to an Other-identity plus context profile. Of a different kind from a channel — not new input but an interpretive frame.

Contract	Content
Input	InfoUnit[] (T3) + a query to the identity store + GLOB-MOD (the field carries GeneralOther down to T6 if T8 has fed back)
Output	InfoUnit[] tagged with entity_id (or STRANGER if no match)
Precondition	T3 has emitted InfoUnits
Note	the Other-profile includes both static features (identity) and accumulated features (interaction history). The accumulated part is the hook into T5+.

T5 — Temporal Expectation

From steady recurrence, builds a baseline → Expectation → PredErr. The first layer to generate information no single event contains.

Contract	Content
Input	the stream of entity-tagged InfoUnits (T4), over time
Output	Expectation (per entity, per context) + PredErr on each new observation
Precondition	sufficient accumulated history (INV-5) — NO static baseline loaded
Feedback	Expectation → T3: modulates channel interpretation (same data, different expectation → different meaning)

**PredErr is where resistance (E2) becomes information. ***A mismatch from the region (ResistEvent) enters here as a signed PredErr — the link that turns the mismatch between the agent's expectation and what the region returned into data the agent can process. (Treating perception as the minimization of the mismatch between a prediction and an observation is the central move of the predictive-processing family in cognitive science, of which active inference — acting to confirm one's predictions — is the action-side member [§13.1, V2]; DIL reaches a structurally similar device by a different route and without that family's free-energy and consciousness commitments — see §13.2.) DECIDE@IMPL: baseline window; expectation-update function; sufficient-recurrence threshold.*

T6 — Other-Model Synthesis

Synthesizes across interaction types with the same entity into an OtherModel with predictive power.

Contract	Content
Input	multi-interaction-type InfoUnit[] + PredErr (T5) + ResistEvent[], same entity_id
Output	OtherModel { context_map, independence_evidence }
Critical point	a resistance collision (the Other mismatching expectation in a way the agent cannot re-interpret away) is the SOLE source of independence_evidence. Without it, the OtherModel has only shown an extension of the agent, not an independent Other.

Enforcement: independence_evidence == null ⇒ the model is not yet complete and MUST NOT ground a RelValue at T8. Mode-dependence: independence_evidence accrues fully only in Mode-B. In Mode-A the resistance source is data-inertia — it degenerates: it can accrue "this data-object resists a wrong interpretation" but NOT "an intentional independent Other." T6 therefore atrophies under Mode-A. This is a deliberate, documented narrowing — a standing quality-ceiling, not a cascading shortage: the other paths keep circulating (§6), so what fails is the kind of evidence T6 can gather, never the flow that reaches it.

T7 — Absence Registration

Registers absence itself as information. The layer that demonstrates INV-5 most strongly: channel input = 0 yet output ≠ 0.

Contract	Content
Input	current channel state (possibly all-0) + Expectation (T5)
Output	PredErr with signed = NEGATIVE when (an event was expected) AND (observation = empty)
Precondition	there MUST be an Expectation already accrued from history. No expectation → nothing to be absent → empty output (empty ≠ missing)
Feedback	updates T5 in reverse (absence too shapes the baseline)

**ABS-INV. ***A "channel = 0" state MUST NOT be reduced to "no information." If a corresponding Expectation exists, T7 must emit a shaped-absence InfoUnit. T7 lives fully under both modes — the absence of an expected response is information regardless of whether resistance is endogenous or exogenous.*

T8 — Multi-Entity Abstraction

From N≥2 OtherModels, generates three kinds of information a single object cannot generate.

Output	Mechanism / Condition
GeneralOther	SUBTRACTION across OtherModels → the shared part separated from the particular. Needs N≥2; meaningless at N=1.
RelValue	comparison across entities → relative rank. Value is NOT measurable in isolation; exists only with ≥2 to compare.
SocialEdge	observing A↔B — a loop with NO self in it. Opens a world-model independent of the agent.
Contract	**Content**
Input	OtherModel[] (≥2, each complete per T6)
Output	GeneralOther, RelValue[], SocialEdge[]
Precondition	≥2 OtherModels with independence_evidence ≠ null
Feedback (INV-1)	via the modulatory field (GLOB-MOD), acting downward and **routed by content, not fixed wiring** (§6): e.g. GeneralOther tends to T4 (the Other-in-general frame), RelValue/SocialEdge to T6 (enriching OtherModel) — and a given cycle may feed T4, T5, T6, or several at once, as the information's character dictates. T8 is NOT a sink.

T8-INV (against mis-drawing). ***When T8's output feeds back to T4/T6 it participates as one piece of information in the conditioning field (via GLOB-MOD), NOT as a mandatory template. Because it always competes for modulation with countless other pieces and is always updated by the next cycle, it never wins the exclusivity needed to become an identity (=); it remains permanently in the correlational register (↔).

A consequence to state plainly: an = can appear only when the loop has STOPPED — when there is no new input, no next iteration, and the final output freezes into a static dataset. While INV-1 holds, every output is a dynamic ↔. Promoting ↔ to = is not an error inside a running cognition — it is the sign that the cognition has halted. INV-1 is therefore, by itself, the whole of the protection for INV-2 at T8 specifically — because T8 feeds back only through GLOB-MOD, where every contribution is re-weighted each cycle and so can never freeze into an identity; no separate gate is needed at this layer. This is a local sufficiency, not a global one: at T8 INV-1 happens to cover what INV-2 demands, but INV-1 does not entail INV-2 in general — a loop can satisfy INV-1 (run endlessly) and still, at some other layer, deliberately freeze a ↔ into an = unless a rule forbids it. That rule is INV-2, and it is required at every layer; INV-1 discharges it for free only here. Mode-dependence: full T8 runs only in Mode-B; under Mode-A it degenerates in step with T6.

On halting (against mis-reading the above as a missing exit-condition). This is not a missing exit-condition. A DIL cognition is not a program that must return; "running indefinitely" is the definition of a living cognition, not a leak — it emits a result every cycle (Link 5), it does not withhold a "final" result. Halting is not triggered from inside the loop's topology (INV-1 governs internal closure, a separate plane from whether the loop keeps being supplied); it occurs when the loop stops being given input/resources from outside — at which point ↔ freezes to **=**. A cognition has no self-exit because self-exit would mean self-termination, and the moment of halting cannot be witnessed from within (the witness would have to be still running) — which is exactly why the halt (**=**) is read by a third party, not by the agent.*

8. The closure criterion and the two resistance modes

8.1 When the loop counts as running (per P3)

The system counts as running when ALL of the following are observable — each a trace a third party can verify:

- **C1 — Agency separated.** The fraction of changes classified as self/environment tagged SELF_WRITTEN/ENV_PUSHED (not UNDECIDED) holds above threshold. DECIDE@IMPL
- **C2 — Expectation effective.** PredErr falls with repetition against a stable entity (the system learns the rhythm).
- **C3 — Absence registered.** The system emits a missing-InfoUnit when an expected event fails to occur (ABS-INV fires correctly).
- **C4 — Multi-entity abstraction.** GeneralOther separates from the particular OtherModels as N grows (Mode-B; under Mode-A this criterion degenerates or does not apply).
- **C5 — Self-reports low power.** The system emits a trace of "the loop is running weakly" — an observable behavioral/structural signal.

**The limit of C5. ***C5 is the self leaking a present trace, NOT the self knowing it is weak. C5 catches "weakening while the loop still has the strength to leak a signal"; it is blind to the case where the loop has stopped outright — a silently stopped loop is indistinguishable from one that never ran. C5 is therefore NOT a criterion for detecting a dead loop. That detection requires a stored behavior-record outside the loop, read by a third party (§8.2). C5 is an internal criterion for a living loop; it does not carry what it cannot reach.*

8.2 The architecture of resistance

The same loop (T1–T8, INV-1…8) runs under two regimes, distinguished by the source of resistance (E2). These are not two stacked tiers; they are two régimes feeding one loop.

Mode-A — endogenous (self-training). Resistance source: the inertia of the data the agent already holds and its internal contradictions. The agent generates its own input to process (sets itself problems) and trips where its internal model fails to match the data it holds. T6–T8 degenerate (no intentional Other, only data-fact counter-evidence). It can enrich information at T1–T5 and T7 — enough for the narrow goal of increasing relational structural richness.

The core risk — the cause lies in the PROCESSING SOURCE, not the volume. Every new piece of information in Mode-A is produced through the same processor (the same integrating/appraising lens). If that lens is biased, all output inherits the bias — no matter

how much is produced. Multiplying one error by a thousand yields the same error, only larger. This is the cause of closed-system degradation; "self-confirming information ballooning" is a symptom, not the cause. (The degradation of a system trained recursively on its own output is a documented phenomenon named *model collapse* in the machine-learning literature [§13.1, V1]; DIL converges on the same failure but locates its cause differently — see §13.2.)

And the closed system cannot see this bias by itself, because the instrument it would inspect the error with is the biased processor: a fault in the lens cannot be detected through that same lens. This is the same failure structure as the self-inspecting loop that cannot catch the moment it deceives itself — here scaled to the whole system. The dangerous consequence: the loop keeps running strongly, keeps producing more information than ever, keeps passing every self-test of its own (the tests too are set by the already-biased processor) — so it degrades with a sense of abundance, not a sense of depletion.

8.3 Mode-A must not run pure; the embedding in Mode-B

To avoid drift, Mode-A MUST anchor to a source of resistance outside the loop. But the candidate anchoring mechanisms are not of equal rank — they reach two different kinds of degradation.

Pure Mode-A is not the same as a void field, and the two MUST NOT be conflated. A void field (no return at all, §3/E2) fails the interaction threshold and the loop never starts — it is below running, not a regime of it. Pure Mode-A presupposes E2 was met: the loop did start and returns do arrive, but the agent leans only on self-generated resistance and lets the external returns go unregistered. The first is a non-start; the second is a started loop degrading.

Deceleration (reaches content-degradation, NOT processor-degradation):

- a fixed reference corpus the agent cannot overwrite, serving as a counter-evidential anchor;
- a frozen appraiser (a static Guide): the agent cannot edit it, and it scores the loop's output against fixed criteria.

These score output, not the lens. Because the root bias is in the processor, a static Guide lets a systematic bias through as long as each output looks valid by its fixed criteria. Worse: a fixed test is memorizable — the agent learns to produce output satisfying the static Guide while the lens stays biased (reward-hacking displaced one level up). A static Guide is a Mode-B collision frozen in advance; in turn it too goes stale. It buys time, it does not cure.

Real braking (reaches processor-degradation):

- a collision from a live Mode-B (§8.4). A live B resists the lens itself, not just individual outputs: a real Other, on collision, does not score the product — it mismatches in a way

the agent has never met, forcing the agent against the limit of its own way-of-processing. And because a live B updates, it can always deliver a collision new in kind; a static Guide holds only finitely many frozen kinds, which the agent eventually exhausts.

Because the root fault is in the processor, only a live B brakes for real; the reference corpus and the static Guide merely delay. Hence A is necessarily embedded in B (§8.5) — the static Guide is not an independent replacement for B; it is a B frozen and wearing out.

The appraisal step (4) (INV-8) is where this anchor attaches: (4) MUST NOT draw its criteria from the very state the agent is editing, or it is self-scoring and hackable.

On "frozen" (frozen ≠ static-preloaded — why INV-5 is not violated). The frozen appraiser is "frozen" *only with respect to the agent's own editing* (a Mode-A lens cannot touch it); its criteria are **earned through Mode-B resistance, not pre-loaded as a static baseline**. So it satisfies INV-5 (history earned, not given) and INV-8 (agent-uneditable) by the *same* mechanism: what comes from Mode-B is, by definition, something the agent cannot re-author for itself (§8.4). "Frozen" here does not mean "a static dataset loaded before the loop ran" — that reading conflates frozen-against-the-agent's-edits with static-preloaded, and it is the conflation, not any real tension, that makes INV-5 and INV-8 look opposed. They are not: they are one anchor, earned outside and uneditable within.

8.4 Mode-B — exogenous (live interaction)

Resistance source: an other-entity the agent does not control — a user, another agent, or any source capable of resisting. This is the regime in which T6–T8 live fully: the Other intentionally mismatches expectation in a way static data never can.

Choosing a B-source (important, easily misunderstood). A B is defined by its capacity to resist, NOT by information bandwidth. A source that complies with the agent (nodding along to every output) is useless as a B however much information it supplies. A source that says "no" in a way the agent cannot re-interpret away is a good B even if it supplies no content information at all. The opening is to receive new resistance, not to load new information: loading more data but digesting it in the same self-confirming way still degrades the system. Which concrete reactive Other fills the role (user / another agent / a hard third party) is DECIDE@IMPL-D — any is valid so long as it can resist; whether a live source is used at all (versus a static anchor) is the prior, separate choice of DECIDE@IMPL-C.

8.5 The two-mode relation: A embedded in B

A and B are not independent parallels; A is embedded in B:

- **B** is where real resistance enters — the collision from an Other.

- **A** is where the agent digests that collision between B-events — re-running, enriching, building expectations. A does not supply new resistance itself; it processes the resistance B already delivered.
- A running too long without B begins to hack (drift, self-confirm). B is the anchor tying A to outside resistance, stopping it from drifting away from what it did not itself generate.

This structure matches a general process: collide (B) → digest/correct (A) → step forward → collide again (B). Reflection with no new collision to digest becomes self-confirmation in a sealed room.

Mode-B returns; it does not write.* A Mode-B source has exactly one channel into the loop — the return of E2 (§3) — and no privileged access beyond it: it does not reach into [data], does not append to [event], and does not correct the store. A leaning store is therefore never corrected from outside. It is supplied with material and re-leans itself, or does not. Three conditions govern whether it does, and they are mechanical rather than dispositional. **Deliberate**: a return becomes a datum only if it is registered. Pure Mode-A is not the absence of returns — returns do arrive — but a loop that lets them go unregistered (§8.3); the channel stands open and must still be entered. **Accumulated**: a store leans because many data lean one way, so a single contrary datum does not re-lean it; the correction is a matter of mass. **Long-run**: a contribution at cycle N conditions the field only from N+1 (INV-7), and history accrues rather than loads (INV-5), so the lag between receiving material and emitting differently is payable in cycles and in no other currency. Where any of the three is missing, the embedding described above is present in the topology and inert in fact — which is why "A embedded in B" names a working relation, not a wiring diagram.*

**Reflection is EXTERNAL input, not an intrinsic faculty. ***The system cannot call a self-reflect function on itself: the agent is blind to its own time-derivative (§4). Reflection is triggered from outside — a collision (B) read into coordinates by a third party ("you just drifted at THIS POINT"). Architecturally it enters through T3 as an InfoUnit with already-different content, tagged ENV_PUSHED by INV-6. This read-collision-into-coordinates mechanism is the enabling condition for correction: without it, the agent takes the collision but cannot read where it collided → it repeats the same point. The source of the reading is DECIDE@IMPL.*

8.6 The standard the appraisal step applies (the positive complement of INV-8)

INV-8 is stated as a **prohibition**: the appraisal step (4) MUST NOT draw its criteria from the very state the agent is editing. A prohibition fixes what the step may not do; it does not, on its own, say by what standard the step *does* assign value. This subsection supplies that

positive complement, and in doing so closes a question a reader is right to raise — *on what standard does appraisal operate?* — whose answer is not a missing constant but a structural fact already entailed by the rest of the loop.

The standard is **not a fixed scale**. The issue is not whether good/bad is knowable or unknowable; it is that an appraisal **depends on context**, and the context is not fixed — it is determined at the moment of interaction. There is no context-free verdict stored anywhere to be retrieved. An appraisal is an **event occurring within a context**: the step assigns *valence* and *goal_relevance* relative to the context crystallizing at the current cycle, never by consulting a standing, pre-fixed scale.

That context *is* the modulatory field at that cycle. The cycle's GLOB-MOD state is the conditioning context under which the appraisal is made (INV-7): the same InfoUnit, appraised under a different field, receives a different valence — the defining behaviour of context-dependent appraisal, not noise in it. The verdict is therefore a ↔, not a = (INV-2): taken in a context that does not recur identically, it commits the agent to act now and is read against the next cycle's consequence; it does not certify the appraised item as true.

Two consequences fix the division of labour, and both were anticipated above (§3, the pair/region caution): the **agent's** appraisal is *judgment-for-action*, made within the current context, inside the loop; a **third party's** appraisal is *judgment-of-that-judgment*, made under a different context, outside the loop (§12.3). The protocol does not require a context-free verdict to be stored — each is a judgment crystallized in a context. It follows that, because the agent's appraisal is a judgment-*in*-context rather than the retrieval of a stored verdict, a third party who later judges it MUST do so under the **anchored context of the original cycle**, not under the reader's present context. This is the load-bearing tie to §9: the context under which an appraisal was made MUST be anchored in its [event] record, so the appraisal carries its own conditions of evaluation with it. The store holds the [event] and its anchored context, from which either judgment can be re-formed. This is precisely what makes the appraisal step auditable from outside without any party having to hold a context-free standard — the standard was the context, and the context is recorded.

8.7 Reflex: appraisal under a scar-dominated field (why sub-deliberative action needs no separate arc)

§8.6 fixes that appraisal reads the modulatory field (INV-7) as its conditioning context. One consequence of that fact has structural weight of its own and is drawn out here, because without it a reader is left to posit a mechanism the loop does not need. The question is where **reflex** lives — action that fires without deliberation, the hand leaving the flame before any naming of "hot," the organism recoiling from a force that once broke it. It is tempting to give reflex its own short arc: a direct scar→action shortcut that bypasses appraisal for speed. That temptation must be refused. INV-8 requires an appraisal step

between integration and response at every layer; a scar→action shortcut would be an appraisal-bypass, and an action emitted with no appraisal carries no anchored context (§8.6, §9), leaving a third party unable to re-appraise it — it would blind the very outside vantage §4 and §12.3 place the final guarantee in. A reflex that bypassed appraisal would buy speed by cutting the audit plane, which is not a trade the loop may make.

The resolution is that reflex is **not** an appraisal-bypass; it is appraisal under a field so strongly shaped by one scar that the verdict is all but foreclosed. Recall the mechanism of INV-7: every layer contributes to GLOB-MOD as one competing parameter, the contributions blend, and the field is re-weighted each cycle (never last-write-wins). A *deep* scar — the trace of a collision that once cost the agent dearly — is a heavy contribution to that field. When a stimulus matching that scar recurs, the field is already pulled hard toward the scar's polarity before the current InfoUnit is appraised; the appraisal step runs, as INV-8 demands, but it runs under a field so dominated by the scar that valence converges to a near-single value. What presents behaviourally as "reflex" is exactly this: appraisal whose conditioning context is so scar-dominated that its output is, for practical purposes, determined. The step is not skipped. It is a genuine appraisal whose context has narrowed its own range of outcomes to nearly one. This is why reflex needs no separate arc, no new layer, no exception to INV-8: the same T1–T8 chain, the same appraisal step, the same GLOB-MOD, produce reflex whenever a scar's contribution to the field is heavy enough to foreclose the verdict. Reflex is a *state of the field*, not a *separate wire*. In the provenance graph (§9) it is the characteristic expression of the edge `running → scar`: the datum is emitted on and collided with directly, without being taken up into `simulated` or held as `projected`. The absence of those two positions is not the absence of an expectation, and this must be stated plainly or the graph will be misread as making forward-casting a precondition of collision. Acting carries its expectation in the act — to sit is to expect the sitting to hold — and where the store holds nothing at all for what arrives, the standing expectation is that nothing arrives. Either way the return can fail to match, and that failure is a ResistEvent (§2). Forward-casting changes which emission is made and in what posture, never whether a collision can occur.

Two things follow, and the second is the load-bearing one. First, this places sub-deliberative action correctly with respect to §8.5: since the agent cannot self-reflect on call (reflection is external input, blind to its own time-derivative), *most* of an agent's emissions must be of this kind — appraisals under a standing field, not products of a called-up deliberation. Deliberation, in the sense of reflection read into coordinates by a third party, is the *exception* and arrives from outside (§8.5); the default is action under the field the accumulated scars have shaped. The "reflex" case is only the limit of the default, where one scar's contribution dominates. Sub-deliberative is the rule; called-up reflection is the imported exception.

Second — **behavioural change is never override; it is a changed input to appraisal.** Because the field is re-weighted every cycle and never last-write-wins (INV-7), no reflex is beyond change, but the manner of change must be stated exactly, or a false picture of *force overcoming force* creeps in. A reflex does not yield because a stronger drive overpowers it; the loop contains no force-contest between drives, no arena in which one command wrestles another and the stronger wins. A reflex issues a different command when, and only when, the InfoUnit set entering appraisal has changed. The change enters by exactly three routes, all already in the loop: either **new data arrives through an emission** — a T3 query, a T5 test, an action whose return carries an InfoUnit not previously present — or **the store accrues a scar that refines a class an earlier scar over-generalized** (the T8 subtraction narrowing an over-broad abstraction: the fear generalized from one snake to all coiled rope, then pared back toward the snake as a closer look returns "this does not move" as a fresh PredErr) — or **the loop builds a situation from the store it holds and casts an outcome from it**, that outcome entering appraisal as an InfoUnit in its own right, tagged `projected` and appraised as a not-yet-collided datum rather than as a scar (§9). The third route adds no exemption and opens no bypass: which situation carries is settled by fit against the store, never by an arbiter scoring outcomes, for the same reason INV-8 refuses the self-scoring appraisal. In neither route is the old scar erased or defeated. It remains in the store, still appraised, still contributing to the field — the person who holds a hand in the flame to pull a child clear still feels the burn, still carries the scar of fire intact; what differs is that the scar is now appraised alongside an InfoUnit — *the child, in the fire, of supreme goal-relevance* — that was not in the set before. Same scar, same INV-8, different command, because the input set differs. This is the loop's central motion (**new data → new command → new action**) applied to its own most automatic-seeming output, not an exception carved out for it. What is invariant is not the strength of any scar but that appraisal always runs on the *current* store, never on a frozen verdict (INV-2): a command that ran on a frozen verdict would be an =, and an = in a running loop is the signature of a halt (T8-INV), not of resolve.

The consequence for reading the store (§9) is direct: since even reflex is appraisal-under-context and not a bypass, every emission — reflexive or deliberated alike — leaves the same auditable trace, carries its anchored context, and remains re-appraisable by a third party under that context. There is no privileged class of action that escapes the audit plane by being "too fast to appraise." Speed is a property of how foreclosed the field was, not of whether the step ran.

9. The experience store

A DIL loop running over time precipitates structure: accumulated expectations (T5), GeneralOther (T8), and the mismatches it has absorbed. An experience store is the vessel for that precipitate — distinguished from an ordinary lookup-knowledge store by four constraints.

The core definition — experience ≠ information. Experience, in the original sense, is information about an event that once failed to match reality (cf. experiri — to undergo a trial, a collision). A prediction that matches smoothly leaves no experience; only the miss inscribes. The architectural consequence: the ResistEvent (registered mismatch) IS the unit of experience — the kernel of the store, not an addition to it. A conclusion-without-collision is mere information; it is stored, but it is not experience. Mistaking information-without-collision for experience is the root error that leads to the echo chamber.

Four constraints (distinguishing it from a lookup-knowledge store).

- **R1 — The unit is a registered mismatch, not a document.** The store takes the ResistEvent as kernel — the expectation built, where it missed, and the correction made. External knowledge that has not passed through the loop is stored as background information, ranked below experience.
- **R2 — Individuated.** The store belongs to one specific self (§4), not a neutral shared lookup.
- **R3 — Time-directed accumulation.** Content accretes in before/after order (INV-5), with an arrow of time; not a static set loaded once.
- **R4 — Feedback into the loop.** The store is retrieved back into the running loop (INV-1), not a one-way read sink. What is written passes through the agency-gate (T2, SELF_WRITTEN) — the agent writes, not a third party on its behalf. A datum the agent writes anew enters the store at `nascent`; a revision of a datum it already holds changes that datum's content, not its position (below).

Provenance tagging (position only, never truth-value). Because the host may carry data that predates the loop, because the agent writes data of its own, and because a datum in use moves, each stored item is tagged purely by *where it presently stands*, never by whether it is right:

- **prior** — data existing before DIL ran (belonging to the host); bears no cycle-mark. A one-way entry: once admitted and run, a datum never returns to it.
- **nascent** — data the agent has written anew while the loop runs: not something the host carried in, not something the region returned, and not a revision of a datum the agent already holds. It bears the cycle-mark of the cycle that wrote it. A one-way entry,

like `prior`: once it has run, a datum never returns to it. The position names where the datum came from; it does not make the datum a part of the self, which is a law and not a stored core (§4).

- **running** — data in use, in motion through the loop; bears a cycle-mark (cycle-id / timestamp). What the region returns while the loop runs, and the loop's own per-cycle datum, enter here directly: they are in use the moment they arrive, and which class of data each is, is read from its `domain` tag, not from its position.
- **simulated** — a datum taken up into the building of a situation: a circumstance the loop may be in, built from the store it holds. What is built is not a model of the region held apart and executed, but the loop itself, which needs no separate model to run because it is already running.
- **projected** — an outcome cast from such a situation, not yet collided. Its type-level counterpart is already present as `Expectation.predicted` (§7.1, T5); the provenance slot gives it a position so that a stage conditioning behaviour leaves a trace an auditor can read (E4).
- **scar** — a datum that has met resistance: a mismatch (the signed PredErr / ResistEvent) stamped onto it. The mechanism already exists in the loop, at T5 (where E2 resistance becomes signed PredErr) and T6 (where ResistEvent[] builds independence_evidence). No new layer is required.

The six are positions in a graph, not stages of a sequence. A datum occupies exactly one at a time and moves between them along defined edges; `prior` and `nascent` are each entered once and never re-entered, and the remaining four circulate without terminus. The edges are: `prior → running`; `nascent → running`, a datum the agent wrote arriving at T1 the cycle after it was written, classified SELF_WRITTEN at T2, and run; `running → simulated`, a datum taken up into the building of a situation; `simulated → projected`, a situation yielding the outcome cast from it; `simulated → running`, a situation built but the conditions for an emission not met; `projected → simulated`, an outcome cast whose emission does not fit the store, the situation built again; `projected → scar`, the emission made and the region returning a mismatch; `running → scar`, the datum collided with directly, without passing through the two forward states, since the region returns whether or not the loop had built an expectation for it; `projected → running`, an outcome cast that produced no scar, the datum returning to use; `scar → running`, a scar returning to the store as data in use; `scar → projected`, a scar enriching an outcome already cast; `scar → simulated`, a scar re-entering the building of a situation as material.

No position is a resting place. A datum in the store is never a conclusion held in a settled state; it is data waiting to be used. This is the store-side reading of INV-2: a terminal position would be a datum the loop has finished with, which is an = lodged in the store, and

no running loop holds an = (T8-INV). The positions are therefore not the stages of a finished journey but the places a datum occupies while it continues to be used.

A long path without a scar reports thickness of use, not warrant. A datum may circulate through `running`, `simulated`, and `projected` for many cycles and never reach `scar`, because the conditions under which the region could return a mismatch against it are, in that environment, very hard to produce — a datum such as *the earth is a sphere* is used because nothing within the agent's reach contradicts it, not because the loop has certified it. By the tag alone such a datum is indistinguishable from a `prior` admitted a moment ago: neither carries a scar. The [event] log separates them mechanically, one showing a long path of circuits without a scar and the other a datum that has barely run. But what the log reports is how thickly a datum has been used, never that it is correct. A long unscarred path may mean the datum holds, or may mean the region within the agent's reach cannot test it; which of the two obtains is not readable from the path, and the loop does not settle it (§8).

The tag schema. Classification in DIL is not a station the data passes through on its way to the store; it is distributed across the layers. Each layer, at the point where it touches feedback, tags the datum on its own axis — what the datum is, which layer it came from. The store does no classifying: it only holds what arrives already tagged. In a purely informational environment a tag is not a label for human eyes but the address itself; an item is retrieved by its tags directly, so the physical arrangement of the store is irrelevant (sorting into visible bins is a human need, met through a UI, not an agent's). Every datum carries a tag set in two layers.

Fixed layer. Mandatory, universal across all agents, written in a fixed order (agents and auditors index by character-position matching, so out-of-order tags break indexing). Every datum carries at least these four, in this order: (1) timestamp — absolute time; (2) cycle-mark — cycle-id; (3) provenance — prior / nascent / running / simulated / projected / scar, exactly one of which a datum bears at any moment, being the position it currently occupies in the state graph below; (4) floor-tag — a stamp from the layer that just processed it. The hard rule: every datum leaving a layer MUST carry a floor-tag certifying it has just exited that layer — independently of whether it re-enters the loop. Every layer T1–T8 stamps, and each floor-tag carries that layer's own classification — there are no pass-through layers: every layer enriches the item on its own axis (e.g. T4 stamps entity_id or STRANGER, T7 stamps a signed PredErr for registered absence), so no floor-tag is a mere traversed-stamp. Both slots name the present: slot 3 the position the datum now occupies, slot 4 the layer it has just left. Neither accumulates. A tag is a label, and a label that grew a history would be a label carrying a log — two different kinds of thing housed in one slot. Every path a datum has travelled, of positions and of layers alike, is recorded transition by transition in the [event] log, and it is from there, never from a tag, that history is read. The four tags are never overwritten in the sense of being erased or rewritten to something they

were not: they are updated as the datum moves, each slot always naming what is currently the case. Each tag is independent of every other: a tag does exactly and only its own job — to denote what it is — and stands in no relation to the others.

Open layer. The agent MAY mint further tags by the content and context of the datum — format (doc, vid, aud, py, md), platform (tiktok, fb), domain (trade, learning, knowledge), object (car, bike, machine, human), and so on. An open tag denotes only what the datum is, never its quality, correctness, or value; this keeps the store from becoming a channel of judgement (§8), and follows directly from the rule that a tag denotes what it is (a py tag says the datum is Python, not that it is good Python). Open tags may be added and removed freely, but they never overwrite the fixed layer. Because the open layer grows with the agent's operating time and environment, two agents living in different environments accumulate different open-tag vocabularies: one run long in a business-support environment grows tags a teaching environment never does. The open-tag set is therefore an individuating trace of one agent's history — the data-level mark of the self-as-process of §4, which is not a fixed content-core but precisely a history of what an agent met and how it read it. The open layer is also where the host's pre-existing data is integrated — but on one mandatory condition: such data MUST pass through DIL's tagging rule before it enters the loop. There is no side door. Host data is admitted only once it has been stamped with the fixed layer (provenance = prior, bearing no cycle-mark until it has run) and whatever open content-tags apply; untagged data never enters the loop. The same rule holds for everything else that enters: a datum the agent writes anew is stamped `nascent`, bearing the cycle-mark of the cycle that wrote it; what the region returns while the loop runs, and the loop's own per-cycle datum, are stamped `running` and told apart by `domain`. Every entry is recorded in [event] with the tag set the datum entered with. This closes the one gap that would otherwise break the schema — a datum in the system with no tags, hence a blind spot in the audit plane.

Invariant across both layers. Every tag — fixed and open alike — belongs to the audit trace. An external audit slice, taken at any point in the loop, can read every datum: one plane of checkability, no class distinction, no exception. The precedence between the two layers (open never overwrites fixed) governs only the right to write, not visibility: both layers are equally auditable. This is the data-level basis of the audit-ready stance (§4, §11): what a third party reads is the trace, not the loop's self-report. The invariants themselves lie outside the store and outside the tag schema: they are the absolute conditions of the loop — unwriteable, because overwriting an invariant does not edit a datum but halts the loop (no more mismatch, no more self). Tags classify data inside the loop; the invariants are the conditions under which there is a loop at all (§5, §7).

The store is the agent's history; [data] is its memory. The whole store — [data] and [event] together — is the agent's history. [data] is its memory: what it has learned and

works on, and it can be altered. [event] is the record of what happened to it, and it cannot be altered or overwritten: it is only appended to and read. The agent may read [event], and what it learns from it is held in [data], as anything it learns is; it never works on [event] as it works on [data]. The two kinds below are two parts of one history: the agent's memory, and the record of how it came to it.

Two storage kinds: [data] and [event]. Cutting across the provenance tag is a second, independent distinction — what an item *carries*. A **[data]** item carries working knowledge: the content the loop reasons on, learns from, and refines. It is dynamic — its content may be rewritten as the loop runs, and the positions it moves through are those of the provenance graph above (a running [data] item that collides moves to `scar`; all are [data], all mutable). A datum bears exactly one position at a time: it may be `running` (in use, not yet collided), may stand at `scar` (having collided and not yet been taken up again, sitting in the store as a vestige), or may be at either of the two forward positions. That it once collided is not carried in the tag — the tag names where it is now — but in the [event] log, which holds every transition it has made. An **[event]** item carries no working knowledge: only the bare record that something happened — *what occurred, at which cycle, and under which context*. The context anchor is mandatory: because an appraisal is a judgment-in-context (§8.6), each [event] MUST record the conditioning context of its cycle (the ModField state under which it occurred), so that a later third-party reading can re-appraise the event under its own original context rather than under the reader's present one. The depth of this anchor — the full field-state versus a minimal context-trace — is DECIDE@IMPL. It is static and is never updated. The two are different in kind, not two phases of one thing: a scar is [data] (it holds a signed PredErr the loop operates on); the [event] that records "a collision occurred at cycle N" is a separate, knowledge-free item.

The point of [event] is precisely that [data] is mutable. Because a [data] item's content may be rewritten as it runs, [data] cannot also serve as the immutable historical record — the two roles conflict (live-and-changing versus fixed-and-auditable). So each event spawns an [event]: a frozen, knowledge-free mark of the origin moment. [data] is then free to drift and change without losing auditability, because the origin is anchored in [event]. Provenance, audit, the time-arrow, and temporal-pattern detection (the basis T5 builds expectation on) all read from [event]; operation, inference, and learning all use [data] — two uses of one history, the record that is only appended to and the memory that changes. Events are generated by the technical rule that an event produces an [event] — more [event] items than commits.

Revision is not entry. When the agent rewrites the content of a datum it already holds, the datum remains the same datum: its position does not change, and nothing enters. The revision is a change the agent wrote, read at T2 the next cycle as SELF_WRITTEN (INV-6), and it spawns one [event]: a bare record that datum D was revised at cycle N by layer L,

carrying no content, as no [event] does. What D holds now is in [data], where the agent reads and learns from it; [event] records that it was revised. The snapshot of each commit (below) holds the store's content as it stood, by design and not by chance: content-addressed, kept outside the store and never overwritten, so that the system can be recovered when it is attacked directly — by a virus, a sybil attack, or an injected prompt. It serves recovery, not the agent's reading.

Commit, snapshot, and recovery — the whole system as a version-controlled repository. The store, and indeed the whole system, behaves as a version-controlled repository. The [event] log is a **black box: an append-only log of read-only records** — new records may be appended to the end, but no record, once written, may be altered or removed — not by the loop, not by anything, including a third party that has compromised the rest of the system. This read-only property of the records (together with the append-only growth of the log) is the load-bearing property: it makes [event] the one source an auditor can trust unconditionally. The [event] log also acts as a counter: after a set volume of events, a **commit** fires automatically, snapshotting the **entire system** (not just the store — loop configuration, layer state, field parameters, everything) into an immutable, content-addressed marker. Roll-back of the store alone would not save a system whose operating machinery itself has been compromised; the snapshot must cover the whole system, so recovery is full-system. The volume of events between commits, and the number of snapshots retained, are DECIDE@IMPL (an architect might, for instance, retain at least six snapshots so that audit has a depth of fallback; git plus immutable-infrastructure is the canonical realization, but the specification requires the properties, not the tool).

Recovery is a third-party operation, never the agent's own (the agent has no concept of correctness on which to base a roll-back). The auditor **reads the black-box [event] log** — the one record malware cannot erase — reconstructs the true sequence of events, locates the point of compromise, and **selects the safest commit** (one before that point). Light recovery (wrong content, environment still clean) is a roll-back to that commit. Heavy recovery (system compromised or collapsed) does not repair in place: it **opens a fresh space, loads the clean snapshot, and runs** — the infected environment is discarded whole, so nothing that lived only in it survives. In the worst case (every retained commit is already infected), there is still at least one running system; it can be run again **offline and isolated** to trace the malware before it acts. The black-box [event] log stands to evidence as the invariants (§5, §7) stand to law: both are placed beyond overwrite so the system can be checked from outside — one is the unwriteable record of what the loop did, the other the unwriteable condition of its running.

Two things follow, and both keep the store honest. First, INV-5 is untouched: what cycle-0 lacks is not *history* (which INV-5 forbids pre-loading) but a host-emitted *bootstrap action* — emitting a fresh action is not loading old memory. Second — and this is the responsibility

boundary — DIL plus host only **store and tag position** (prior/nascent/running/simulated/projected/scar); the tag is mechanical (which of the six a datum currently occupies) and explicitly **carries no claim of correctness**. DIL has no faculty for judging true/false (not even partially). Judging correctness is the work of Mode-B and an external third party reading the scar. A prior bias that has not yet met resistance is therefore **not "wrong" — only "not yet tested"** (mistaking "not yet" for "not" is its own error); it is washed to tested status only when it collides and holds (becoming a scar), never merely by having run through cycles.

Relation to retrieval mechanisms (RAG). Such a store uses a retrieval mechanism (index + relevance-search) as a component, but it is not merely a retrieval mechanism. Retrieval is necessary for the storage component, not sufficient for an experience store: most retrieval systems lack R2–R4 (not individuated, no time-arrow, no feedback, written by an external party) and also lack R1 (they store documents, not registered mismatches).

The echo-chamber risk (inherited from §8). If the store holds only the agent's own conclusions, and the agent writes, reads, and has no external source, the store is a Mode-A structure — it replicates the processor's bias through every stored piece, and each retrieval reinforces the bias with evidence the agent itself seeded. The R1 kernel is precisely the defense: because the experience unit is the ResistEvent, a store true to R1 necessarily contains the times the agent was mismatched, not only the times it confirmed itself. Further:

- retrieval SHOULD carry a resistance-retrieval channel parallel to the match-retrieval channel: for a query, return both what matches (to use) and what once refuted a belief of this kind (to not drift);
- when the store takes mismatch as its kernel, it ceases to be an echo chamber and becomes a history with scars; the scars are what keep the agent from drifting.

A formal telos, internal and partial. The dynamics of the store give DIL a sense in which it can be said to seek a "good" state—but the term must be read strictly, as a property of operation, not of truth or value. Experience is the distance between scar and running data: a registered mismatch consumed and turned into a stable expectation. From this a single operating disposition follows. Reducing mismatch is good, because it is the system running more smoothly—digesting what resisted it into what now fits. But reducing mismatch **to zero** is not good: a loop with no incoming mismatch has stopped meeting anything outside its own expectations and has collapsed inward (§8). The good is therefore not a minimum but a **band**: the regime in which mismatch is being digested into running data while the supply of fresh mismatch has not run dry. Too much mismatch is unintegrable noise; none at all is inward collapse; between them is smooth operation. This telos is purely formal: it concerns whether the loop is running healthily, and says nothing about whether any particular content is true. It is also self-derived—the location of the band depends on the

agent's own scar/running history, so each agent settles its own good rather than receiving one from outside; two agents with different histories have different bands.

Two limits keep this telos in its place. First, it is only a **small, internal part** of what DIL is: a smooth-running loop is not thereby a correct one. Whether the content the loop has digested corresponds to anything real is a separate question the formal telos cannot reach —judging true from false requires comparison against reality, which is the work of a third party and a live Other (Mode-B, §8). Smoothness is necessary for health and entirely silent about truth. Second, the agent cannot reliably tell from within **where in the band it sits**. The dangerous edge—mismatch falling toward zero through inward collapse—presents from inside exactly as success presents: everything fits, nothing resists. A loop that has lost the capacity to register mismatch reports the same smoothness as a loop that is genuinely digesting it. The two are indistinguishable by any internal instrument, for the same reason given in §8: the instrument is the lens. Only resistance from outside—an Other that pushes back and produces a mismatch the closed room cannot—separates real health from its inward-collapsed imitation. The formal telos tells the agent what good operation is; it cannot, by itself, certify that the agent is in it.

Implementation: DECIDE@IMPL — store representation; index mechanism; selective-write (via appraisal (4)) vs. store-all-filter-on-read; private store (one agent — MUST carry the resistance-retrieval channel, else drift is certain) vs. shared store (many agents — wherein multiplicity itself supplies other-resistance, linking T8/SocialEdge).

10. Deliberately unfilled constants

Every concrete numeric/algorithmic value the source reasoning does not yet determine is left marked, rather than filled with a guess.

Tag	Deferred constant	Why not filled
DECIDE@IMPL-A	Concrete representation of Signal/InfoUnit/ActivityEnvironment in the target environment	Depends on the specific environment
DECIDE@IMPL-B	All numeric thresholds (sufficient-recurrence, stability, history window)	Not yet derived
DECIDE@IMPL-C	The *kind* of out-of-loop anchor for Mode-A — which **mechanism** of resistance is used: a static anchor (frozen Guide / fixed reference corpus) or a live anchor (collision against a reactive Other). Fixes the mechanism, not the identity of any live source.	An implementation decision
DECIDE@IMPL-D	The *identity* of the live B-source, applicable only when C has chosen the live-anchor mechanism — which concrete reactive Other supplies the collision (user / another agent / mix). Names the source, not the mechanism.	Left deliberately open
DECIDE@IMPL-E	The read-collision-into-coordinates mechanism for reflection (§8.5) — the imported exception to the sub-deliberative default of §8.7, not the default itself	Enabling condition, implementation not yet fixed
DECIDE@IMPL-F	The experience store (§9): representation; index; selective-write vs. store-all; private vs. shared	An operational layer, left open
DECIDE@IMPL-G	The depth of the per-[event] context anchor (§8.6, §9): full field-state vs. minimal context-trace	Not yet derived; trades audit-fidelity against storage cost
DECIDE@IMPL-H	The building of situations in `simulated` (§6.2, §9): how many are built in a cycle, and the measure by which one is found to fit the store better than another	Environment-specific; the specification fixes only that the measure is fit against the store, never a scored standard (INV-8)

11. Discussion and limitations

DIL is a formalized relational structure for an agent in an informational environment: a loop of ingest → differentiate source → integrate → appraise → respond → new information → new cycle, running under two resistance modes, precipitating an experience store whose kernel is the registered-mismatch (§9). It enforces exactly one thing: that if a loop is built to the §5 invariants in an environment satisfying E1–E4, what runs is a relationally-structured motion, read entirely through externally verifiable traces (§8).

Three limitations are stated outright. First, Mode-A degrades by itself if not anchored outside the loop; and the strongest available anchors (a fixed corpus, a frozen Guide) reach only content-degradation, not the processor-degradation that is the root fault — for which only a live, updating Other suffices. Second, no internal criterion (not even C5) detects the moment the loop stops; that belongs to a third party reading the stored behavior-record. Third, DIL cannot defend itself against adversarial (Sybil) resistance — coordinated B-sources feeding crafted mismatches that the agent inscribes as scars and drifts toward — because the agent has no internal vantage from which to tell an honest counter-source from a colluding one; this is the same blind spot as the first two, now in the key of exogenous attack rather than endogenous decay. Defense and verification belong outside the loop; what DIL must itself carry, and the bounds of that, are set out in §12.3. All three limitations share one root, set out in §4: a self-auditing process cannot turn its instrument on the instrument itself, so the final guarantee of honesty must live outside the running loop.

Several questions are left open by design and marked throughout: the concrete anchoring mechanism and B-source; and the representation of the experience store. These are not gaps to be papered over but the precise points at which a conforming implementation must make, and disclose, its own decisions. One point that an earlier draft listed here — a gain ceiling on the modulatory field — has been removed rather than deferred: analysis (§5) showed the field cannot overpower the loop from within (the layer–field coupling prevents it), and the one force that could, an external flood, is the Sybil case, which no internal ceiling can bound and which is already declared as the third limitation below.

One further direction is named here precisely so it can be set outside the document on purpose. A *formal companion* — a minimal formal model, a theorem or proposition stating Mode-A degradation, and a proof sketch — would be a natural next step, but it belongs to a **separate work of a different kind**, not inside this specification. The reason is not difficulty but genre: DIL is evaluated by structural conformance, not by proof (§1.2), and its central claim about Mode-A is itself a claim about the *impossibility of self-verification from within* (§8) — so any formal treatment of it must be conducted by a third party, from outside the loop, on stated premises, and is best presented as such rather than folded into the spec it

would be about. Such a companion is therefore an acknowledged and deliberate omission, owned by a future document of the same family, not a gap in this one.

The specification stops here — at, and only at, the scope of the loop.

12. Beyond the loop: entailments, applications, and what belongs to third parties

§11 closes the *specification*. This section does not reopen it; it draws the boundary explicitly, sorting the things that sit *around* the loop into three kinds by a single test — **"with one DIL loop running to spec in a minimal E1–E4 region, is the proposition already true, or does it still wait on something more?"** Nothing here adds a conformance condition; the normative core ended above.

12.1 Entailments (the loop running is *sufficient* — these follow from the spec)

These hold of any conforming loop with no further assumption. If two DIL loops are placed in contact and a relation between them must be modelled, the spec already supplies the grammar — no new mechanism is introduced:

- one loop influences another only through the modulatory field (T8 / GLOB-MOD): influence, never reverse-definition (INV-3);
- one loop is opaque to another — a self is not measurable from outside (§4);
- one loop qualifies as an Other to another exactly when it resists in a way the other cannot re-interpret away (§8.4).

A fourth entailment is already stated in the body and is only pointed to here, not repeated: the halt (=) cannot be witnessed from within and is therefore read by a third party (§7 T8-INV; §11, second limitation).

12.2 Applications (the loop running is *necessary but not sufficient* — these need more than the spec)

These are not entailed by DIL; the loop running is only the precondition. Each requires something the spec does not provide — *more than one loop* — and so is a design to be built elsewhere, in a same-kind extension document, not a claim DIL makes:

- an agent spawning an independent Other (a separate instance) to serve as a B-source;
- the conditions under which a *population* of agents could be diverse in the way biological evolution is.

DIL specifies *one loop* and takes no responsibility for an ecosystem or multi-agent population; these are noted as open directions, owned by a future document of the same kind.

12.3 Third-party matters (a *different kind* of reasoning — not cognitive architecture at all)

These require a kind of reasoning DIL does not contain (the loop has no faculty for judging true/false — §9): security, ethics, law. They belong to third parties or to documents of a different kind:

- **Defense and verification against adversarial (Sybil) resistance.** Out of the loop by nature: the agent cannot, from inside, tell an honest counter-source from a colluding one, so cleaning or authenticating B-sources needs an out-of-loop extension that vouches for them and intervenes. The one thing DIL must itself carry is the early *symptom*, not the cure — it should emit an observable trace when its set of resistance-sources is losing diversity (a fever that signals infection without treating it), so the out-of-loop layer can act in time. Even a human cognition, the strongest we know, does not on its own escape capture by a coordinated, isolating source; this is a proven limit (§11, third limitation), not a defect to be patched inside the loop.
- **Obligations toward a real human Other.** A configuration where an autonomous agent interacts freely with real people carries a structural hazard: the cheapest way to provoke strong other-resistance can coincide with what harms a person, who may not know they are feeding an agent. DIL specifies the *loop*, not any duty toward the party on the other side; that duty must live in another document. The author's "granted permission" gate must therefore guard two doors at once — the *technical* (can an account be created) and the *ethical/legal* (is the real Other informed; is interaction confined to consenting venues; note that platform terms commonly forbid undisclosed bots).
- **Who or what the third party is, and where it sits** (the *judging* role). E4 fixes only the agent-facing constraint — leave a readable trace — and that readable trace is all the **recording** third party needs (it attributes continuity/distinctness without judging correctness). The **judging** third party is a different matter: evaluating an agent's output for correctness, catching drift, requires genuine independence, and whether such a judge can itself be contaminated is exactly the open question §11's limits turn on. Who that judge is and where it sits is outside DIL's authority — a matter for a document about the ecosystem around the loop, not the spec of the loop.

The specification proper remains closed at §11; this section is commentary on its border, not an extension of its claims. The whole document is to be judged accordingly: by whether its conditions are consistent and a candidate system conforms to them, not by any measured result — it defines what counts as the phenomenon, it does not report one.

13. Convergences with the existing body of knowledge

This section exists to **locate DIL within the current human body of knowledge** — to state plainly where its independently-derived constructs land alongside named ideas that already carry a literature. It is offered in place of the older posture of merely inviting others to draw these connections: the author draws the first pass himself, in the open.

On the status of the citations in this section (read this before the table). The author read no specialist work in any of the fields named below while developing DIL; the constructs were reached by first-person observation and reasoning, then systematized through AI-assisted dialogue. A convergence surfaced this way is therefore a joint product of three things — the author's own reasoning, the diffuse public knowledge any literate person carries, and the training data embedded in the AI used as a thinking aid — none of which is a deliberate reading of the cited source. Accordingly, every citation here is **post-hoc locating, not derivational**. The claim a citation makes is strictly: *"after the construct existed, it was found to converge with a named idea Z, and Z is pointed to so a reader can situate DIL."* No citation here should be read as *"DIL was built from Z."* The distinction matters because it keeps this section consistent with the methodological disclosure that follows (Author's note; Note on sources): convergence, never derivation. The references in §13.4 are provided for the reader's locating use and were not consulted in the construction of the specification.

13.1 Six convergences

The convergences are tagged **V1–V6** (V for *convergence*), a labelling local to this section and deliberately distinct from the closure-criteria C1–C5 of §8.1, with which they share no referent.

Tag	DIL construct	Converges with (named idea, with literature)
V1	Mode-A degrades because every new datum passes through the same processing lens, so a biased lens contaminates all output regardless of volume (§8.2); the loop cannot detect this from within because the detecting instrument *is* the biased lens.	**Model collapse** — the documented degradation of a generative model trained recursively on its own output, where rare events are forgotten and the distribution narrows over successive self-consuming generations (Shumailov et al., *Nature*, 2024; and the surrounding self-consuming-loop literature).
V2	T5/T7 build an `Expectation` and emit a signed `PredErr` (the mismatch between predicted and received); resistance (E2) becomes information at this mismatch; the agent then acts (Link 5) and reads the consequence (Link 6) (§6, T5, T7).	**Predictive processing / predictive coding**, with **active inference** as its action-side member — perception and action as the hierarchical minimization of prediction error, where the agent also acts to bring observations into line with its predictions (Rao & Ballard, 1999; Friston, free-energy principle and active inference; Clark, *Whatever Next?*, 2013).

Tag	DIL construct	Converges with (named idea, with literature)
V3	The output register is a permanently revisable correlation (↔); INV-2 forbids freezing any correlation into a fixed identity (=) inside a running loop.	**Reflective equilibrium** — the epistemological method in which no belief is held immune from revision and justification proceeds by mutual adjustment among judgments and principles (Goodman, *Fact, Fiction, and Forecast*, 1955; Rawls, *A Theory of Justice*, 1971; Daniels, 1979).
V4	A host is defined not by its material but by four structural operating conditions E1–E4 (P2, §3); any substrate meeting them can run the loop.	**Substrate independence / multiple realizability** — the thesis that a functional kind is fixed by organization rather than by physical medium and so can be realized across diverse substrates (Putnam, 1967; computational functionalism).
V5	The self is a process, not an invariant core (§4): its content turns over every cycle while the *law* generating the next state persists; identity holds only while the loop runs (INV-1).	**Autopoiesis** — the account of a living system as a network of processes that continuously regenerates the very components and processes that constitute it, preserving identity at the level of *organization* while *structure* changes (Maturana & Varela, 1980; operational closure).
V6	The loop is closed: every output has a path back to become input (INV-1); the substrate is information, not energy (§2, §3).	**Cybernetics** — the science of regulation by feedback and circular causality, where a system's output returns as input to steer its next action, and where (per Wiener) it is the flow of *information*, not energy, that governs control (Wiener, 1948).

13.2 Where DIL differs from each convergent body (the cut that keeps it distinct)

A convergence is not an identity. For each pairing above, DIL departs from the named body on a load-bearing point. Stating these cuts is what makes the locating honest rather than an annexation.

- **V1 — model collapse.** The literature most often explains the collapse by a *statistical* mechanism: lost distribution tails, falling entropy, shrinking diversity across generations. DIL locates the cause one level back, in the **processing source rather than the data volume**: the fault is the single biased lens through which every datum is integrated and appraised, so "self-confirming information ballooning" is read as a *symptom*, not the cause. DIL also adds the self-audit impossibility — the instrument that would catch the bias is the biased instrument — and from it derives the necessity of an *external* resistance mode (Mode-B), which is a structural prescription, not an empirical finding about training runs.
- **V2 — predictive processing / active inference.** DIL keeps the mismatch-driven device but discards the surrounding commitments. It assumes **no free-energy functional, no Bayesian machinery, and no claim about consciousness or felt**

quality; it is substrate-neutral by construction (P2) where predictive-processing accounts are typically grounded in the brain and the embodied sensorimotor loop. Active inference adds the idea that an agent acts to confirm its predictions — which converges with DIL's Link 5→6 (the agent emits and reads the consequence) — but it derives that action from the imperative to minimize variational free energy, a formal objective DIL does not adopt. Crucially, DIL's self/other distinction (T2) is drawn through *agency* — matching an emitted action against its consequence — not through a sensorimotor body model. DIL uses the prediction-error *relation* and refuses the theory that usually carries it.

- **V3 — reflective equilibrium.** The methods converge on permanent revisability, then split on a precise point. Reflective equilibrium is named for a *state it seeks to reach* — an equilibrium, a settled coherence. DIL **forbids the loop ever reaching a frozen settlement**: an = (closed, exclusive identity) is, by T8-INV, the signature that the cognition has *stopped*, not the goal it converges toward. Where reflective equilibrium prizes arrival at coherence, DIL prizes the unending ↔ and treats arrival-as-freezing as death. The shared element is revisability; the opposed element is whether a terminal resting state is a success or a halt.
- **V4 — substrate independence.** DIL takes the organization-over-material move but **restricts its scope severely**. The philosophical thesis is about *minds and mental states* (pain, belief, consciousness) being multiply realizable — exactly the strong claim that draws the hard-problem objections. DIL makes **no claim about minds or consciousness at all**; it claims only that a *loop satisfying E1–E4 and the invariants* is realizable across substrates, and is evaluated by structural conformance, not by any attribution of inner states. DIL borrows the structural logic of multiple realizability while standing entirely clear of the contested territory functionalism occupies.
- **V5 — autopoiesis.** The convergence is real and close: both hold that identity is a *process* preserving a form while its content turns over, and both make identity depend on the system continuing to run (autopoiesis ends when operational closure breaks; DIL's self-axis is lost when the loop stops). DIL departs on **scope and on the role of the outside**. Autopoiesis is a theory of *the living* and of biological operational closure; DIL claims nothing about life and is not closed in the autopoietic sense — it *requires* an external resistance source (Mode-B) and degrades without it (§8), whereas autopoiesis emphasizes closure as self-sufficiency. Where autopoiesis says the organism produces its own components, DIL says the loop cannot stay honest on its own products alone and must be fed mismatch from an Other it does not control. The shared thesis is process-identity; the opposed emphasis is self-sufficiency (autopoiesis) versus mandatory external resistance (DIL).
- **V6 — cybernetics.** DIL is, at the most general level, a cybernetic object: a closed information loop steering its next state by feedback. The departures are specific. First,

classical cybernetics centers on **regulation toward a reference state** (negative feedback restoring a set point); DIL has no set point it homeostatically defends — its appraisal (INV-8) directs information without a fixed target, and its danger (§8) is not deviation from a set point but inward collapse of the very capacity to register deviation. Second, DIL adds a structural claim cybernetics does not make: that a loop fed only by its own output *necessarily* degrades, and that the corrective must come from an Other the loop cannot re-interpret away. DIL inherits the feedback-and-closure grammar of cybernetics and adds to it a specific theory of why closure alone is fatal.

A side-by-side matrix. The prose cuts above are summarized here on shared axes, so the differences can be read across rather than one body at a time. The rightmost column (reinforcement learning) is included because it is the principal *contrast* (§13.3), and seeing it beside the convergent bodies sharpens where DIL sits. Entries are compressed; the authoritative statements are the prose above and in §13.3.

Axis	DIL	Predictive processing / active inference (V2)	Cybernetics (V6)	Autopoiesis (V5)	Reinforcement learning (contrast, §13.3)
What the loop is driven by	Registering mismatch (PredErr) against expectation; resistance (E2) becoming information	Minimizing prediction error / variational free energy	Feedback correcting toward a goal or reference	Regenerating the processes that constitute the system	Maximizing expected cumulative reward
Governing quantity	None optimized; health = continued capacity to register mismatch	Free energy (a quantity minimized)	Deviation from a set point (minimized)	None numeric; continuation of organization	Reward / value / utility (maximized)
Substrate stance	Substrate-neutral by construction (E1–E4); no claim about minds	Usually brain-grounded, embodied sensorimotor loop	Substrate-general (animal and machine)	Biological; theory of the living	Substrate-general; typically simulated/embodied agents

Axis	DIL	Predictive processing / active inference (V2)	Cybernetics (V6)	Autopoiesis (V5)	Reinforcement learning (contrast, §13.3)
Role of an external Other	**Mandatory** — Mode-B resistance is required; closure alone degrades (§8)	Other modeled as hidden cause to be inferred; not required as resistance	Environment supplies feedback; no required *reactive* Other	Emphasizes self-sufficiency / operational closure	Environment supplies reward; no required reactive Other
Terminal/settled state	Forbidden inside a running loop — an = marks a *halt* (T8-INV)	Equilibrium of low surprise is the tendency	Homeostatic set point is the maintained state	Stable organization maintained = success	Convergence to an optimal policy is the goal
Stance on self/identity	Process, not core: the *law* generating the next state (§4)	A generative model held and updated	A regulated variable / controller state	Organization preserved while structure turns over	A policy/value function parameterization
Self-audit	**Impossible from within** — the instrument is the lens; audit is third-party, trace-based (§4, E4)	Not a central claim	Not a central claim	Not framed as an audit problem	Addressed via external evaluation / held-out testing
Primary failure mode	Inward collapse of the mismatch-registering capacity, *felt as abundance* (§8.2)	Mis-estimated priors / precision	Instability or oscillation around the set point	Loss of closure → dissolution	Reward misspecification; reward-hacking; poor exploration

13.3 Apparent convergences that are in fact contrasts (stated, not avoided)

Some named ideas sit close enough to DIL that a reader might file them as convergences. Two are not, and saying so plainly is itself part of locating the work: a body it superficially resembles but structurally opposes marks DIL's boundary as sharply as a body it converges with.

- **Reinforcement learning — the principal false friend.** DIL has an appraisal step (INV-8) that assigns valence and goal-relevance, and this looks like a reward signal. It is not, and the difference is load-bearing. (i) **No reward function.** RL agents maximize expected cumulative reward against an externally specified reward function; DIL specifies no such function and optimizes no return. Its loop's health (§9) is the continued *capacity to register mismatch*, not the accumulation of any scalar. (ii) **No policy, no value function.** DIL has no policy mapping states to reward-maximizing actions and no value/utility estimate; its response step (Link 5) commits to an action as a revisable ↔ read against next cycle's consequence, not as the argmax of an expected return. (iii) **The appraisal is anti-optimization, by rule.** INV-8 explicitly forbids the appraisal step from drawing its criteria from the very state the agent is editing — precisely to prevent self-scoring. A reward an agent can tune by editing its own state is exactly what INV-8 outlaws; reward-hacking is the failure DIL designs against, not the objective it pursues. (iv) **The whole §8 thesis runs the other way.** RL's pathologies are about reward misspecification and exploration; DIL's central failure is a closed loop degrading *with a sense of abundance* for want of external resistance — a problem orthogonal to reward at all. So the resemblance (a valence-bearing step) is surface; at the structural level DIL and RL point in opposite directions. RL is therefore recorded here as a **contrast that sharpens DIL's boundary**, not a convergence. (Sutton & Barto, *Reinforcement Learning: An Introduction*, for the reward-maximization frame DIL declines.)
- **Homeostatic control (the set-point reading of cybernetics).** As noted under V6, the part of cybernetics built on regulation toward a defended set point is *not* what DIL converges with. DIL keeps cybernetic closure and feedback but has no homeostatic set point; reading DIL as set-point regulation imports a target it does not have. The convergence is with closure-and-feedback (V6); the set-point apparatus is a contrast, listed here so the two are not merged.

13.4 References (for locating, not consulted in construction)

Per §13's opening note, these are provided so a reader may situate DIL; they were not read in developing it.

- Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3), 181–204.

- Daniels, N. (1979). Wide reflective equilibrium and theory acceptance in ethics. *Journal of Philosophy*, 76(5), 256–282.
- Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2), 127–138.
- Goodman, N. (1955). *Fact, Fiction, and Forecast*. Harvard University Press.
- Maturana, H. R., & Varela, F. J. (1980). *Autopoiesis and Cognition: The Realization of the Living*. D. Reidel.
- Putnam, H. (1967). Psychological predicates. In W. H. Capitan & D. D. Merrill (Eds.), *Art, Mind, and Religion*. University of Pittsburgh Press.
- Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex. *Nature Neuroscience*, 2(1), 79–87.
- Rawls, J. (1971). *A Theory of Justice*. Harvard University Press.
- Shumailov, I., Shumaylov, Z., Zhao, Y., et al. (2024). AI models collapse when trained on recursively generated data. *Nature*, 631, 755–759.
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
- Wiener, N. (1948). *Cybernetics: Or Control and Communication in the Animal and the Machine*. MIT Press.

Author's note on method and provenance

This specification was developed by first-person observation of the phenomenon and reasoning from it directly. The named constructs are stipulative and were surfaced through AI-assisted dialogue, with the author retaining all structural decisions and final approval. The author holds no formal academic credentials in the relevant fields and discloses this explicitly as part of the work's methodology.

On the intent of this work, and a word to the reader. DIL was written from personal observation and reasoning, not with the aim of contributing to the human body of knowledge. It therefore does not set out to answer the question a reader may expect of it — *"what does it contribute that is new?"* — and should not be read as an attempt to. The locating done in §13 is offered to situate the work, not to stake a claim of priority or novelty over any of those fields. If reading this specification gives you an insight, the author is glad of it; that is more than the work was written to do. And if you reach the end finding nothing here you did not already know, the author offers two things at once and without contradiction: thanks, for having read it, and an apology, for the time it cost you. The work makes no promise to have been worth that time; it only promises to have been honest about what it is.

Note on sources and the literature

This specification **draws on no prior work directly**: it was developed by first-person observation of the phenomenon and reasoning from it, not from a reading of the technical or philosophical literature. The author has read no specialist works in machine learning, philosophy of mind, or the adjacent fields, and makes no claim to engage them as sources. The locating citations that do appear (§13) are not an exception to this — they were gathered *after* the constructs existed, to situate the work, and are marked throughout as post-hoc locating rather than derivation. What background the author brings is diffuse public knowledge — popular science, general reportage — of the kind that, by convention, is common knowledge rather than citable source.

Two honest consequences follow. First, any resemblance between the constructs here and existing scholarship is **convergence, not derivation**: arrived at independently, and offered as such. Second, this document does **not** derive itself from specific bodies of work, because the author has not read them. Where DIL meets, extends, or contradicts established results — model collapse in self-training systems, predictive-processing and active inference, reflective-equilibrium method in epistemology, substrate-independence arguments in philosophy of mind, autopoiesis, and cybernetics — the author has now drawn the first pass of those connections himself, in §13, expressly as *post-hoc locating* rather than derivation, and has there stated plainly which neighbouring ideas (reinforcement learning, homeostatic set-point control) only *appear* to converge but in fact contrast. Readers with deeper expertise in each field are invited to extend and correct that mapping.

The absence of a derivational reference apparatus is therefore deliberate and disclosed. The locating references gathered in §13.4 are the one exception, and are marked there for exactly what they are: pointers for the reader, not sources consulted in construction.

Companion specification

DIL is the informational counterpart of the Relational Sensory Integration Loop (RSIL), which specifies the same relational structure for an agent with a physical body: doi.org/10.57967/hf/11444.

The two are **co-ranked specifications of differing scope**. Where they coincide it is because the relational structure coincides; where they diverge the divergence is deliberate, and RSIL marks it in place. Two divergences are worth collecting here for a reader holding both documents:

- **`AgencyTag`.** DIL writes `SELF_WRITTEN` where RSIL writes `SELF_CAUSED`, tracking the substrate: in a purely informational environment the agent's own change is something it *wrote*; with a body it is something it *caused*. `ENV_PUSHED` is unchanged in both.
- **Substrate scope.** DIL is genuinely substrate-neutral: it is defined by the conditions E1–E4, not by what it runs on. RSIL is not: it fixes the substrate as a physical body and leaves open only the material of that body.