ARCHITECTURE

TorinAI Unified Cognitive Substrate

Architecture of persistent learning, reasoning, memory, action, and epistemic governance

Stefan Ragland

Dominion Labs Research & Development

Dominion Labs

research@dmnlabs.org

March 18, 2025

Abstract

TorinAI is a persistent cognitive substrate designed to maintain and develop structured knowledge, memory, beliefs, learned operators, competence, and goals over time. Rather than treating intelligence as the output of a single model invocation, Torin organises cognition as a set of interacting faculties operating over one shared persistent state. Reasoning changes beliefs; experience produces learning; learning produces operators and knowledge; competence shapes exploration; goals drive planning and action; action is independently re-observed; and epistemic governance controls when internally generated conclusions may become authoritative knowledge or executable decisions. TorinAI does not use a language model in its cognition: its persistent state, reasoning, learning, planning, verification, and epistemic authority run on the substrate's own mechanisms. The only component that can consult a language model at all is a single teaching module, and in practice it is essentially unused. This document describes that architecture — its persistent state, its cognitive faculties, the loops that connect them, its grounded model of task completion, and the governance boundary that decides what becomes authoritative.

1 What TorinAI is

TorinAI is a *persistent cognitive substrate*. A conventional model answers a prompt and forgets; the substrate instead holds and develops a body of structured cognition — what it knows, what it has experienced, what it believes and how strongly, what operations it has learned to perform, what it is competent at, and what it is trying to do — and it carries that state forward across time. Intelligence, in this design, is not a single act of generation but the continuing interaction of cognitive faculties over shared, durable state.

The substrate is organised as one coordinator in which each faculty is a pipeline that reads and writes the same persistent state. The faculties are reasoning, learning, memory, execution, domain modelling, intrinsic motivation, and a model of the system's own state. What makes them a substrate rather than a toolbox is that the output of one becomes structured input to another: a conclusion revises a belief, an action produces an experience, an experience induces an operator, a competence gap raises the motivation to explore. The document that follows is the architecture of that interaction.

2 The core architectural principle

One principle organises the whole design: **persistent shared state connects cognitive faculties over time.** Four commitments follow from it.

- **Persistent.** Knowledge, beliefs, operators, and competence are durable; the system accumulates and revises them rather than regenerating from scratch.
- **Shared.** There is one authority for each kind of state — one learning authority, one domain authority, one belief store, one concept graph — read and written by every faculty, not copied per caller.
- **Substrate-first.** Reasoning, learning, and execution run on the substrate's own mechanisms first; where a step can be proven or an operator applied, it is, deterministically.
- **No model in the cognitive loop.** The substrate owns state, reasoning, learning, verification, and authority; no language model participates in cognition. The one component that can consult a language model is a single teaching module, essentially unused (§13).

3 The unified architecture

Figure 1 shows the shape of a full cognitive act, from an input or an internal experience to a completed — and independently verified — outcome that becomes the next experience.

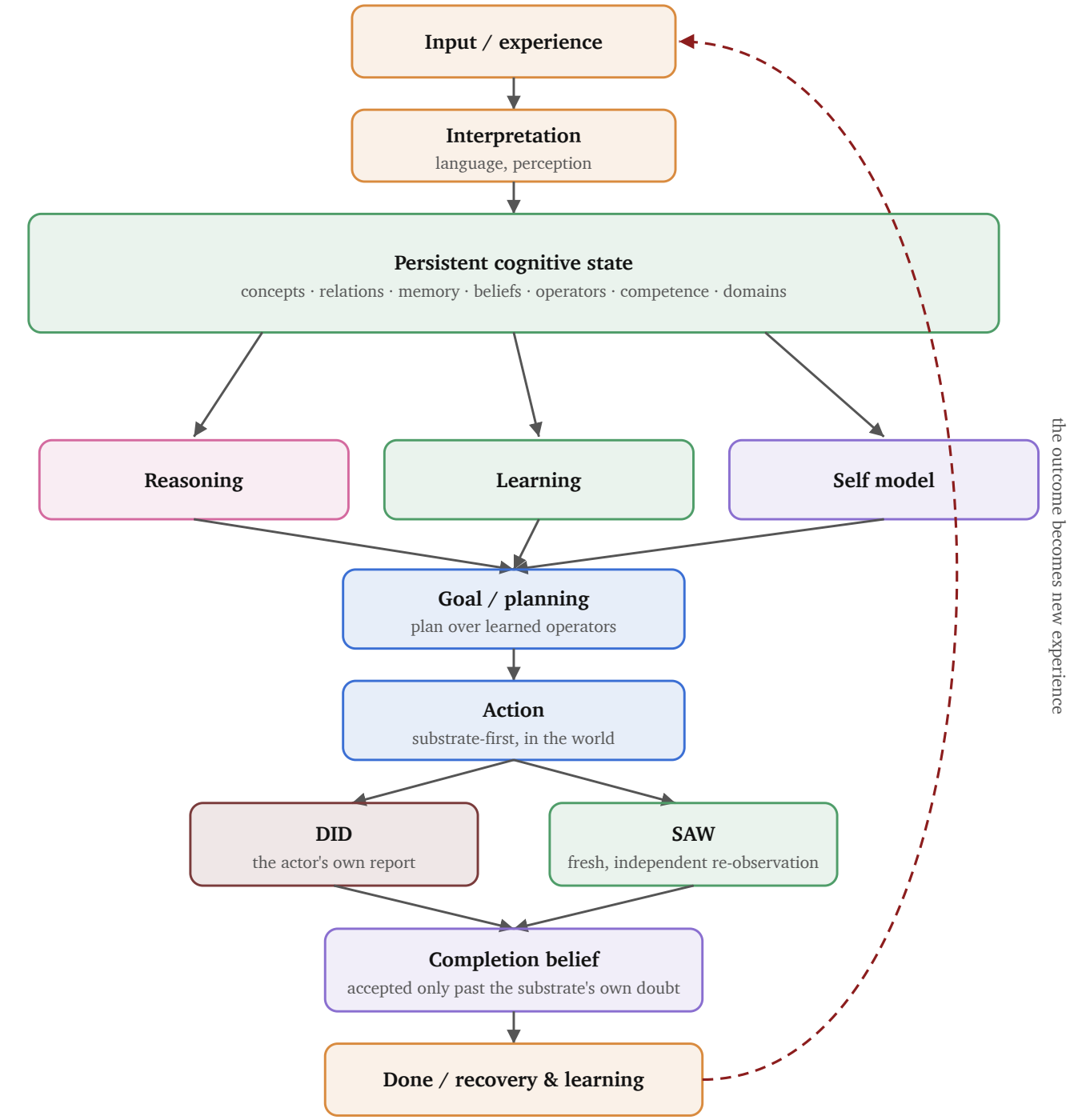

Figure 1. Input and experience are interpreted into the shared persistent state; reasoning, learning, and the self-model read and revise it; goals plan over learned operators; action runs in the world; completion is decided by combining the actor's own report (DID) with a fresh, independent re-observation (SAW), accepted only past the substrate's own doubt; and the outcome — done, or a recovery that produces learning — becomes new experience. The state in the centre is shared by every faculty.

Read top to bottom, this is one pass; read as a whole, it is a loop — every outcome re-enters as experience, so the system's later behaviour is shaped by its earlier acts. The sections that follow define the persistent state at the centre and then each faculty in turn.

4 Persistent cognitive state

The persistent state is the substrate's memory of itself and its world. Its elements are distinct and each has an authority.

- **Concepts.** The units of what the system knows — the nodes of a typed concept graph, each a thing the system can reason about.
- **Relations.** Typed, polarity-bearing edges between concepts (*is-a*, *part-of*, denials, and more); how the graph encodes structure rather than a bag of facts.
- **Memory.** Retained experiences — what happened, what was said, what an action produced — stored when worth keeping and retrievable by content and relation, then consolidated and abstracted over time.
- **Beliefs.** Propositions the system currently accepts, each held with a calibrated uncertainty and revisable as evidence arrives (§10).
- **Operators.** Generalisable, variable-carrying transformations the system has learned — the actions it can take and the derivations it can perform, with explicit add and remove effects.
- **Competence.** A first-class estimate, per subject, of what the system can actually *do* — distinct from how much it knows (§7).
- **Domains.** Subjects — coherent clusters of concepts and operators — created automatically as knowledge accumulates, so cognition is organised rather than flat.
- **Experience.** Action–outcome traces: the before-state, the action, and the re-observed after-state that feed learning.

These are not independent stores; they form a cycle (Figure 2). Experience is retained as memory; memory feeds learning; learning writes operators, concepts, and beliefs; those drive reasoning and planning; planning drives action; action produces the next experience.

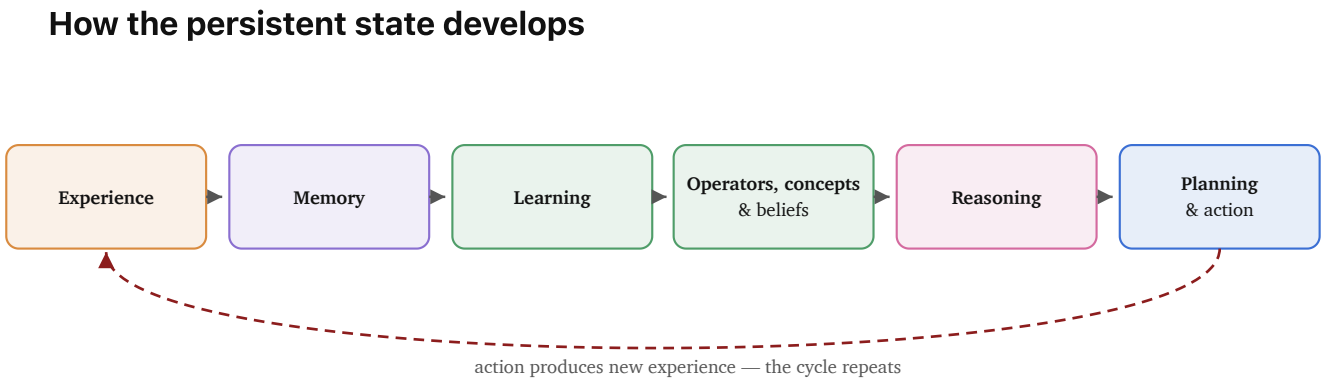

Figure 2. The state is not a set of shelves but a cycle. Learning is the hinge: it turns retained experience into the operators, concepts, and beliefs that reasoning and planning consume, and action closes the loop by producing fresh experience.

5 Reasoning

Reasoning answers a question from the substrate's own knowledge. A natural-language query is first *formalised* into a logical query; the reasoner then selects how to answer it and consults the appropriate mechanism, substrate-first and with no model fallback:

- **Deduction** over held rules and premises;
- **Taxonomic reasoning** over the concept graph (an *is-a* query walks typed relations to a verdict);
- **Rule chaining**, composing held conditionals over held facts to reach a multi-step conclusion;
- **Analogy and cross-domain transfer**, mapping structure from one subject onto another.

Two behaviours are architectural, not incidental. When the reasoner meets a query whose answer is not in vocabulary, it performs **gap diagnosis** — distinguishing a missing fact (acquire it) from a missing operation (practise it). And when it cannot ground an answer it returns **unsupported** rather than a plausible guess: a knowledge-grounded reasoner refuses where a generator would confabulate [4].

6 Learning

Learning is the one path by which the system changes. It has four modes, all writing the same shared state through one authority.

- **Instruction — knowledge acquisition.** A fact or rule, admitted with provenance [18], becomes a concept, a relation, and where appropriate a belief.

- **Experience — operator induction.** From before/action/after demonstrations, the system induces a general, variable-carrying operator — least-general generalisation over the positives, with the preconditions the negatives force into the body [12]. Induction runs off the acting path, so acting stays cheap and learning is the deliberate, always-online half.
- **Consolidation — generalisation and organisation.** Related memories cluster into higher-level structure; subjects crystallise into domains; redundant knowledge is merged.
- **Belief revision.** New evidence moves a posterior rather than overwriting a fact; what was believed can change as the world does (§10).

A learned operator does not become a capability the moment it is induced. It is a candidate until independent experience confirms it; only then is it promoted to executable and allowed to act (§11). This is the point at which knowledge becomes competence.

7 Knowledge versus competence

The substrate tracks two different kinds of mastery over a subject and keeps them orthogonal (Figure 3): **declarative knowledge** — how much it knows — and **procedural competence** — what it can actually perform. Conflating them is a failure mode: a missing fact and a missing operation call for opposite responses, one closed by acquisition and the other by practice. Because the two axes are measured independently, the system can diagnose *which* kind of gap it faces and abstain honestly on that basis — a boundary a single-channel generator does not represent [14]. This distinction is treated in depth in a companion study.

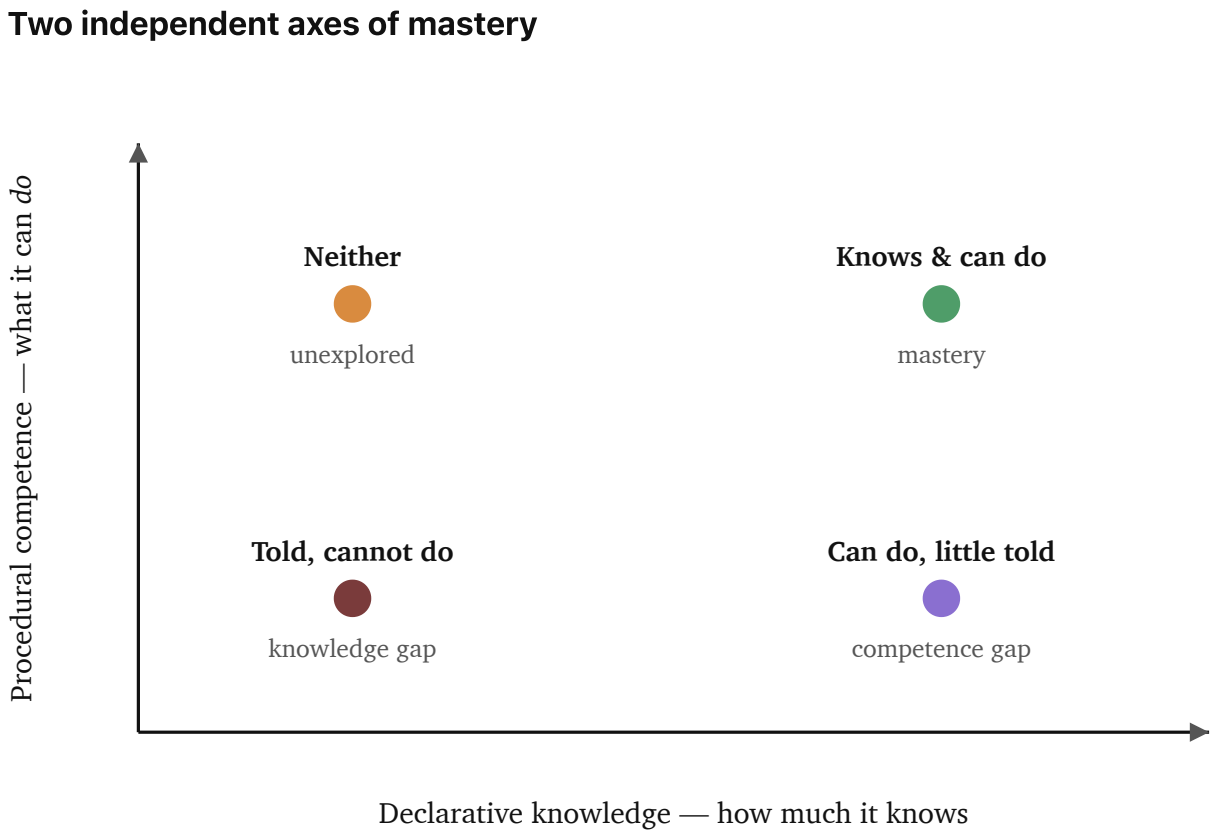

Figure 3. Knowing a fact and being able to act are separate properties. Teaching raises the horizontal axis; practising an operation raises the vertical. The system reads both, so "I was never told this" is never mistaken for "I cannot do this." See Knowing a Fact vs. Being Able to Act.

8 Goal-directed action

Action is planned over learned operators, not scripted. Given a goal state, the execution faculty diagnoses the gap between the current world and the goal, plans a sequence of learned operators that would close it, and drives each step. Where a step can already be proven it runs deterministically; where the goal is a question it is answered through the reasoning loop. A step that fails is not a dead end: the reason routes the response — a missing method is retried by a different one, an unreachable goal means the plan was wrong and is redesigned, a blocker outside the system is surfaced honestly. Every attempt, success or failure, becomes experience that learning can consume; validated experience can update operators and competence, so future performance may change as a result of prior action.

9 Grounded task completion

Deciding that a task is *done* is a cognitive act in its own right, and one the substrate does not delegate to a success flag. Completion is a belief the system forms about the world and must accept past its own doubt. It

rests on two epistemically distinct kinds of evidence, named for readability rather than borrowed from any other field. **DID** (“did”) is execution-side evidence: the action reported producing the intended effect. **SAW** (“saw”) is observation-side evidence: a fresh, independent re-observation of the world confirms the intended effect actually holds, without reusing the actor's report. These fail independently: a tool can report success while changing nothing, and independent observation is what catches it. Completion combines the two under an independence rule and is accepted only when the resulting posterior clears the system's standing confidence bar; the high bar itself forces the fresh observation. Because the confirming channel is independent of the actor, a task cannot be marked done on the strength of the same channel that performed — or faked — the action. This architecture is developed, with measurements, in a companion study: Task Completion as an Internal, Grounded Judgment.

10 Belief and epistemic state

Beliefs are how the substrate holds uncertain propositions. Each belief has a **prior**, is moved by **evidence** to a **posterior**, and carries the **provenance** of what supports it. The update respects two properties that keep confidence honest. **Independence:** confirmations that share a source or a causal lineage collapse to one before they compound, so correlated evidence cannot be counted many times — a single grounding leaves a belief calibrated, not certain. **Reversal:** a belief is revisable, so a later observation that contradicts it moves the posterior back rather than being ignored. This is the machinery behind grounded completion and behind the governance boundary of the next section.

11 Systemic epistemic governance

The substrate continuously derives conclusions, induces rules, and forms beliefs. Not all of them should be allowed to change what the system treats as authoritative or to trigger an action. Governance draws that boundary (Figure 4): a proposition may be *derived* and held in soft cognition — reasoned over, reinforced, generalised from — while being prohibited from becoming authoritative, or from acting, until it passes independent, non-circular validation. This separates the ability to derive a proposition from the authority to persist and act on it, and it is what keeps an internal error from escalating itself into authority.

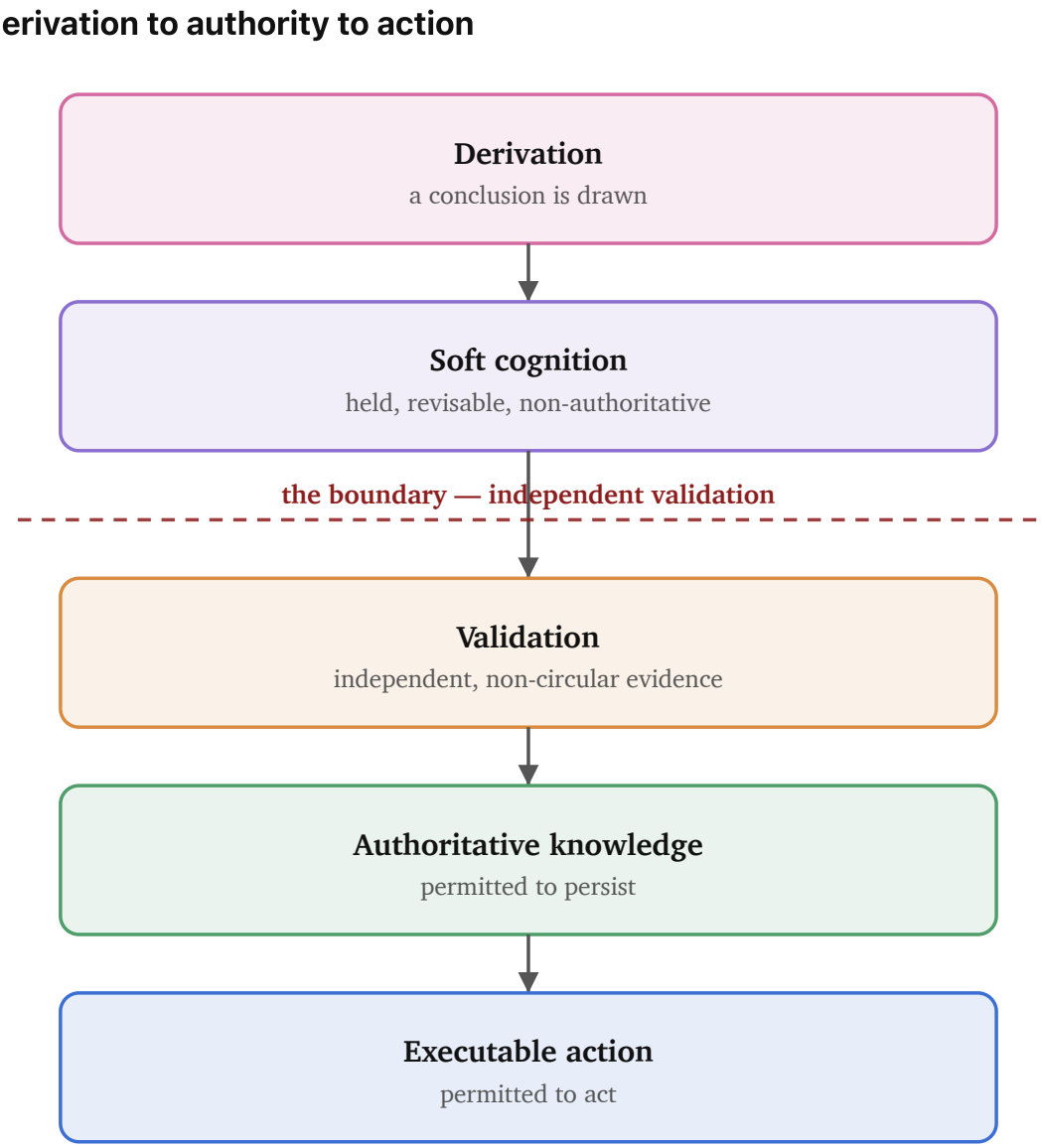

Figure 4. The governance boundary. Derivations live in a soft, revisable tier and are never returned as authoritative or allowed to act until validated on evidence independent of their own derivation — the point marked "the boundary." The formal semantics and guarantees are given in Systemic Epistemic Governance.

12 Cross-faculty loops

The faculties are connected by loops in which each one's output is another's input. Three are load-bearing.

The experience–learning loop. Action produces an outcome; the outcome, re-observed, is retained as experience; experience induces operators and revises knowledge; the updated operators and knowledge shape the next plan. Competence grows from doing.

The completion loop. Action yields DID; an independent SAW is taken; the two form a completion belief; the verdict is done, or a recovery that itself produces experience (§9).

The intrinsic-learning loop. A competence gap raises an epistemic uncertainty; the uncertainty surfaces as an exploration target; exploring produces experience; experience updates the competence belief; the uncertainty falls and the subject stops attracting attention (Figure 5). Attention thus follows what is uncertain and learnable [9],[10], not a fixed agenda.

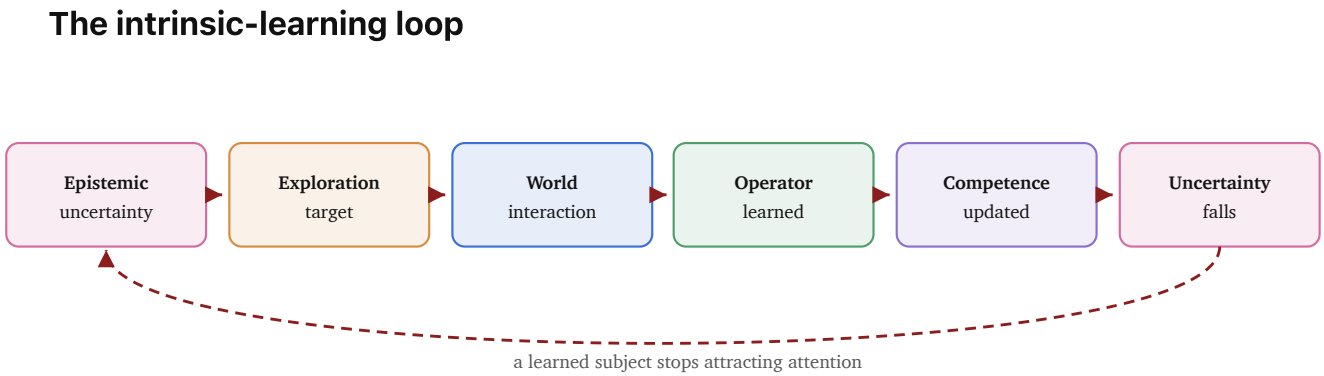

Figure 5. Motivation is downstream of epistemic uncertainty and upstream of learning; learning reduces the uncertainty that initiated the exploration. The loop closes across four faculties — motivation, execution, learning, and beliefs.

13 Coordination, and the place of language models

Coordination. The faculties run in one coordinator over one shared state. A reactive event spine lets one faculty's outcome wake another — a demonstration triggers induction, an admitted fact triggers domain crystallisation, a health event triggers recovery — so the system is event-driven rather than a fixed script. Alongside the reactive spine, a band of always-on disciplines runs on one scheduler: memory consolidation, self-review, knowledge refresh, security, health and recovery, and a constitutional self-check. Each kind of state has a single authority, so there is one owner for beliefs, one for domains, one for learning.

Language models. No language model is part of the cognitive loop: the substrate reasons, learns, plans, acts, and verifies on its own mechanisms, and the paths described in this document consult none. The one component that can consult a language model at all is a single teaching module — a teacher policy that may propose a candidate lesson the substrate then tests against the world — and in practice it is essentially unused. A model never owns state, never decides what is true, and never grants authority; remove it entirely and the substrate is unchanged.

14 Current implementation and validation

Scale. The current implementation is a persistent system of roughly 224,000 lines across some 295 core modules, over a persistent graph of about 175,000 concepts and 238,000 relations, with several thousand revisable beliefs and 21 subjects — several of which the system organised for itself. Scale is context, not the claim; the architecture above is the claim.

Validation. The individual mechanisms of this architecture are studied, model-free, in companion research: relational operator induction with a causal ablation (Learning Relational Action Rules Without a Model); reading as a derived program (A Reading Is a Program over a Sequence); the knowledge/competence distinction (Knowing a Fact vs. Being Able to Act); grounded completion (Task Completion as an Internal, Grounded Judgment); and the governance boundary (Systemic Epistemic Governance). That the faculties described here run together as one system, over one shared state, is checked in situ against the running substrate rather than asserted — summarised in Appendix A.

Appendix A · Architecture verification

The architecture is not only described but exercised: the current system is booted and each faculty's pipeline is driven through the one coordinator in situ, recording present behaviour rather than a stored result. Reasoning answered a taxonomic query from its own knowledge and refused an ungrounded one; execution ran a task and accepted it only after re-observing the world; a belief moved from a low prior to a calibrated posterior on a single grounding rather than saturating; the domain authority held its subjects; motivation surfaced targets from measured uncertainty; memory stored and retrieved. Every pipeline ran, and did so with no language model consulted where a model is not the point. Wiredness is checked against live call sites and scheduler registrations, so a capability counts as part of the running system only where the running system actually reaches it; experimental or deprecated interfaces are not considered part of the canonical substrate unless they are connected to the active architecture.

References

[1] J. E. Laird, A. Newell, P. S. Rosenbloom. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33(1), 1987.

[2] J. R. Anderson et al. An integrated theory of the mind. *Psychological Review*, 111(4), 2004.

[3] J. E. Laird, C. Lebiere, P. S. Rosenbloom. A standard model of the mind. *AI Magazine*, 38(4), 2017.

[4] P. Wang. *Non-Axiomatic Logic: A Model of Intelligent Reasoning*. World Scientific, 2013.

[5] B. Goertzel. *Engineering General Intelligence* (CogPrime). Atlantis Press, 2014.

[6] B. Goertzel et al. OpenCog Hyperon. *arXiv:2310.18318*, 2023.

[7] J. Bach. Modeling motivation in MicroPsi 2. In *AGI*, 2015.

[8] P.-Y. Oudeyer, F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1, 2007.

[9] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation. *IEEE Trans. Autonomous Mental Development*, 2(3), 2010.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[11] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4), 1991.

[12] R. E. Fikes, N. J. Nilsson. STRIPS. *Artificial Intelligence*, 2(3–4), 1971.

[13] G. Ryle. *The Concept of Mind*. Hutchinson, 1949.

[14] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3), 1979.

[15] J. Cheney, L. Chiticariu, W.-C. Tan. Provenance in databases. *Foundations and Trends in Databases*, 1(4), 2009.

[16] M. Minsky. *The Society of Mind*. Simon & Schuster, 1986.