Hyperon

The Open-Source Infrastructure for Artificial General Intelligence

A Deep-Dive Whitepaper on PRIMUS, OmegaClaw+Self+Hive,

and the Path from AGI toward Beneficial ASI

Ben Goertzel

July 2026

Abstract

Hyperon is an open-source framework for constructing Artificial General Intelligence through the coordinated interaction of multiple cognitive paradigms. Rather than treating intelligence as the product of a single learning algorithm or monolithic neural model, Hyperon provides a common substrate in which symbolic reasoning, probabilistic inference, attention allocation, evolutionary learning, neural computation, perception, action, and self-modification can participate in an integrated cognitive process. Its architecture comprises five closely connected layers: MeTTa, a reflective programming language designed for cognitive computation; Atomspaces, dynamic metagraphs representing declarative, procedural, attentional, and experiential knowledge; a collection of interoperable AI algorithms including PLN, ECAN, MOSES, WILLIAM, predictive coding, and related methods; PRIMUS, a cognitive architecture organizing these components into a goal-directed and continually learning system; and ASI:Chain, a runtime environment supporting secure, verifiable, distributed, and potentially decentralized cognitive computation.

A central principle of Hyperon is cognitive synergy. Different cognitive processes have different strengths, weaknesses, representations, and characteristic timescales. Their combination can be substantially more capable than any one process operating alone when their intermediate results are expressed in a shared, inspectable form and their interaction is governed by effective attention, learning, and control mechanisms. MeTTa and Atomspace provide this common language and representational medium, while PRIMUS specifies how reasoning, learning, memory, motivation, creativity, perception, and action may be composed into coherent cognitive systems.

OmegaClaw provides an operational control fabric for deploying this architecture in present-day agentic systems. It represents cognitive capabilities as interchangeable Module Spaces; maintains structured Context Frames for complex tasks; allocates resources through fast and slow attention processes; decomposes goals into dynamically revisable plans; and coordinates local, remote, neural, symbolic, and human-mediated processes. The same interfaces can initially wrap large language models, code executors, databases, sensors, and other existing services, while increasingly capable Hyperon and PRIMUS components are introduced through benchmarking, shadow execution, controlled promotion, and rollback. This enables an incremental path from useful contemporary AI agents toward more deeply integrated cognitive architectures without requiring accumulated knowledge, skills, workflows, or operational infrastructure to be discarded.

OmegaSelf supplies an evidence-grounded reflective and governance plane for this control fabric. It treats an agent's self-model not as a privileged identity object or free-form autobiography, but as a continuously tested, context-sensitive theory about the running system. Immutable observations, provenance-complete evidence closures, contextual self-beliefs, pre-action predictions, discrepancy signals, runtime identity anchors, and directed continuity records allow an agent to reason about what it can do, what it is committed to, how it reached its present state, and how proposed

changes may affect its capabilities and governing constraints. Probabilistic reasoning estimates likely outcomes, while a separately authorized policy layer determines which exact action proposals may be executed. This separation between evidence, inference, governance, and effects provides a basis for inspectable and fail-safe self-modification.

OmegaHive extends these ideas to cooperative populations of agents. Individual and shared Atomspaces, explicit task ownership, structured inter-agent communication, provenance-aware artifact production, capability and permission boundaries, collective self-models, quantitative evaluation, and human-accessible recovery paths provide both a practical environment for multi-agent research and engineering and an experimental platform for studying collective cognition. The hive architecture permits specialized agents and cognitive modules to cooperate without reducing their interaction to unstructured conversational exchange.

Hyperon also provides several complementary routes for integrating contemporary neural networks. Neural models may operate as external Module Spaces; neural computation may be implemented more natively within metagraph substrates; or open-source transformer models may serve as high-capacity perceptual and generative kernels beneath predictive-coding and symbolic-head layers. In the latter approach, symbolic interfaces read from and write to Atomspace, predictive-coding networks support local error-driven adaptation, sparse residual structures provide modular and causally auditable specialization, and selectively looped transformer computation can expose additional inference-time depth. These mechanisms offer a practical bridge between the capabilities of large pretrained neural models and the structured reasoning, continual learning, provenance, and cognitive control required by a broader AGI architecture.

World modeling is treated here as a fundamental architectural function rather than as one special-purpose algorithm. A PRIMUS world model is a braided, multi-rate structure spanning immutable evidence, persistent entities and relations, spatiotemporal maps, uncertain rules, structured associative hypervectors, dense predictive dynamics, motives, options, programs, and transfer maps. Its semantics are action-first: an internal representation is useful to the extent that it predicts how the world, the agent, and the representation itself will change under external and cognitive actions. OmegaClaw Context Frames hold task-relevant slices of this world model, Module Spaces expose its heterogeneous prediction and memory processes, and OmegaSelf applies the same predict–execute–observe–revise discipline to the agent's own capabilities and self-modifications.

Recent work also refines several of the control and reasoning mechanisms operating over this world model. TECAN generalizes ECAN from scalar importance to typed, token-gated cognitive metabolism. Budgeted STLM pattern mining uses dependency-tower compilation to decide which candidate structures merit expensive inference. Semantic algorithmic chemistry evolves and ablates context-sensitive networks of graph rewrites. The context-indexed progression from πPLN to ωPLN keeps persistent positive and negative evidence separate from temporary local Bayesian charts, while adding explicit provenance, context translation, reversible residuals, and path-dependence diagnostics. PLN proof DAGs can also be compiled into procedural generators for hypothetical witness-worlds, connecting uncertain reasoning directly to counterfactual imagination and model-based action selection.

Taken together, the components described here constitute an extensible operating framework for integrative cognition: a reflective language, a shared metagraph representation, a pluralistic suite of learning and reasoning processes, an organizing cognitive architecture, an executable agent-control fabric, evidence-grounded self-modeling, cooperative multi-agent organization, and infrastructure for secure distributed execution. The resulting design is intended not as a fixed blueprint for intelligence, but as an evolvable foundation on which increasingly general, capable, transparent, and beneficial cognitive systems can be built.

Scope note. *This whitepaper presents the Hyperon architecture through its account of Atomspaces, MORK, the MeTTa language stack, PRIMUS, weakness, geodesic control, TransWeave, MetaMo, SubRep, WILLIAM, predictive coding, self-modification, and decentralized execution. It includes the public five-layer framing of Hyperon, the OmegaClaw control fabric, OmegaSelf as an evidence-grounded self-model and governance layer, the OmegaHive multi-agent experimentation architecture, and a concrete bridge from open-source transformers to predictive-coding and symbolic-head cognition. The architecture further incorporates action-first, multi-rate world modeling as a fundamental PRIMUS and OmegaClaw function; HMH-mediated structured associative memory; typed cost-accounted attention; context-indexed and reversible PLN; budgeted STLM pattern mining; semantic algorithmic chemistry; and constructive compilation of PLN proofs into procedural world-generators.*

Claim and evidence note. This whitepaper presents its technical thesis and substantive architecture material using a consistent claim-status vocabulary, explicit mathematical assumptions, reproducibility requirements, cryptographic and recovery scope boundaries, and primary-source references. Numerical figures that are not accompanied here by a complete benchmark package are identified as preliminary engineering observations or illustrative targets rather than silently presented as externally validated results.

Contents

1 Executive Summary

1.1 Core Vision

Hyperon starts from a simple observation: a generally intelligent system will need more than one kind of cognition. High-capacity neural models are powerful at perception, generation, and statistical pattern completion. Probabilistic reasoning is useful when conclusions must be traceable and uncertainty must be propagated. Evolutionary program learning can discover compact procedures. Attention systems allocate scarce computation. Motivational and planning systems decide what should be pursued. Pattern mining and concept formation create reusable structure. None of these mechanisms, by itself, is a complete account of human-level general intelligence.

Most contemporary AI architectures respond to this plurality in one of two ways. One approach places a large neural network at the center and adds retrieval, tools, verifiers, agents, or external memories around it. Another approach builds a hybrid system whose subsystems exchange queries, text summaries, or narrow API objects. Both approaches can be useful, but both tend to make the interfaces between cognitive processes into semantic bottlenecks. The reasoning system cannot easily inspect the intermediate state of the learner; the learner cannot directly reuse the proof structure discovered by the reasoner; the attention system cannot allocate resources over all of them using a common currency; and the system's own control logic is usually outside the representational world it is trying to understand.

Hyperon takes a different route. It provides a common language, a common metagraph memory, a family of interoperable cognitive algorithms, a cognitive architecture, and a runtime capable of spanning one process, a cluster, or a decentralized network. The goal is not merely to connect heterogeneous components, but to let them share structured intermediate states, uncertainty, motives, procedures, attention values, provenance, and control rules. Hyperon is therefore best understood as an operating system for cognition: not an operating system that merely isolates processes, but one that helps diverse cognitive processes cooperate when cooperation is useful and remain modular when it is not.

1.2 Hyperon in Five Interconnected Layers

The public Hyperon framing describes five interconnected layers[1]. They are not a strict software stack in which each layer calls only the layer below it. They are mutually reinforcing views of one system.

1. **MeTTa Programming Language.** MeTTa is Hyperon's native language of thought: a compact language for metagraph pattern matching, rewriting, reflection, and composition. MeTTa programs can represent not only domain procedures but also the rules that select, inspect, and modify procedures.

2. **Knowledge Representations.** Hyperon stores declarative knowledge, procedural knowledge, goals, uncertainty, attention values, neural features, and control state in living hypergraphs called Atomspaces. Atomspaces may be backed by different engines and distributed across different physical substrates while preserving a common Space interface.

3. **Hyperon AI Algorithms.** Algorithms such as Probabilistic Logic Networks (PLN), Economic Attention Networks (ECAN), and MOSES evolutionary program learning operate over the shared substrate. Newer work adds WILLIAM pattern mining and compression, predictive-coding dynamics, SubRep, MetaMo, TransWeave, geodesic control, weakness-based regularization, and other mechanisms described in this paper.

4. **Cognitive Architecture and Research.** PRIMUS configures the algorithms into a unified cognitive system. It specifies how goals are maintained, how tasks are decomposed, how reasoning and learning cooperate, how attention is allocated, how ambient cognition continues between explicit tasks, and how self-modification is constrained and evaluated.

5. **ASI:Chain Runtime Environment.** ASI:Chain provides an AI-native, capability-secured execution substrate in which MeTTa can be compiled through MeTTa-IL to Rholang, cognitive state can be committed to RSpace, and selected reasoning or governance processes can be made persistent and cryptographically verifiable. The same conceptual system may also run privately or on trusted clusters; decentralization is a deployment mode rather than a requirement for every cognitive operation.

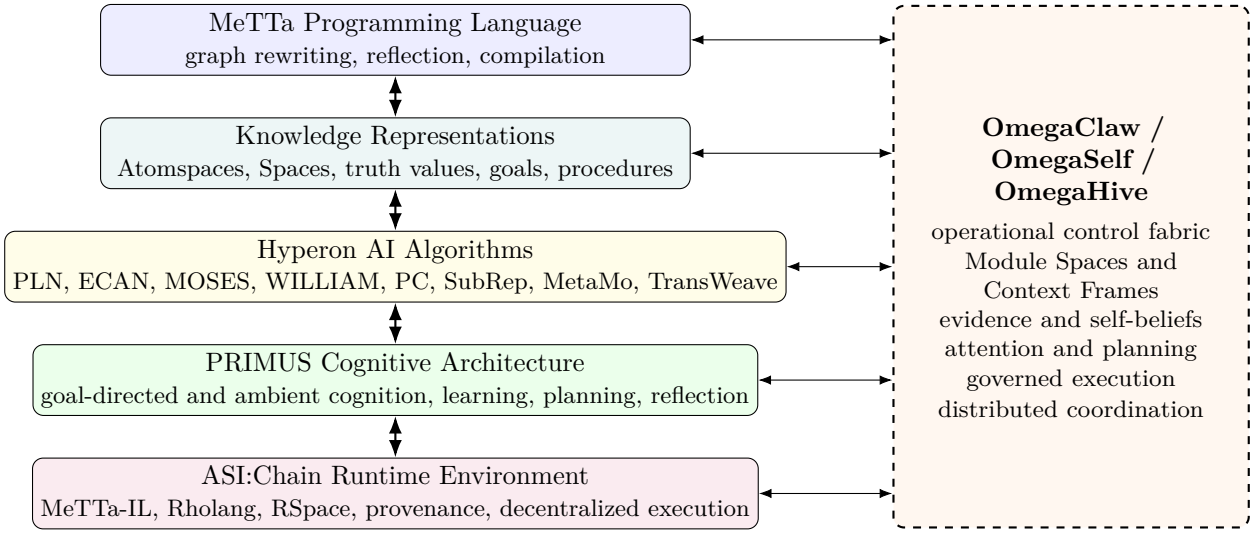

Figure 1: The five public layers of Hyperon, with OmegaClaw, OmegaSelf, and OmegaHive shown as an operational control fabric spanning rather than replacing them.

The five-layer view clarifies a distinction that is easy to blur. PRIMUS is the cognitive architecture: it describes the organization of reasoning, learning, motivation, attention, and self-improvement. OmegaClaw is an operational control fabric: it instantiates, routes, monitors, and governs cognitive work across concrete modules and machines. OmegaSelf is the evidence-grounded self-theory and action-governance protocol threaded through that fabric. OmegaHive generalizes the operational ideas to a cooperative fleet. ASI:Chain is the verifiable runtime and coordination substrate on which selected parts of the system may execute. These roles overlap in implementation, but separating them conceptually helps the architecture evolve without turning into one monolithic code base.

1.3 World Modeling as the Missing Middle

A generally intelligent agent needs more than perception on one side and planning on the other. It needs a persistent, revisable account of entities, situations, causal tendencies, affordances, norms, action outcomes, and its own position in an evolving world. This paper therefore treats world modeling as a fundamental architectural braid running through PRIMUS, OmegaClaw, OmegaSelf, and the application domains, rather than as one neural network or one symbolic database[15, 16].

The action-first criterion is simple. Let R be the procedure that constructs an internal representation of a situation, and let M_a be the internal modeling procedure associated with action a. A useful world model should make

$$M_a(R(w_t)) \approx R(w_{t+1})$$

when action a takes the world from w_t to w_{t+1}. Actions include ordinary external actions, such as moving a robot arm or clicking a web page, and cognitive actions, such as looking again, retrieving a memory, running a reasoner, or testing a hypothesis. On this view, state is best understood operationally as a structured distribution of affordances: what can be done here, under which constraints, at what cost and risk, and with what predicted result.

PRIMUS implements this criterion using three coupled representational regimes. Explicit Atoms and programs preserve identities, roles, rules, norms, and sharp combinatorial constraints. Hierarchical modular hypervectors (HMH) provide fast structured associative retrieval over episodes, skills, affordances, social scripts, and mined templates. Dense vectors and tensors support perception, predictive dynamics, and low-latency action. Evidence anchoring keeps abstractions linked to the sensory or execution records from which they arose. Explicit grounding and lifting operators move information between these regimes, and multi-rate control keeps fast servo or reflex loops independent of slower symbolic consolidation.

OmegaClaw gives this world model an operational form. A Context Frame is a typed, task-relevant view of the current situation, goals, active hypotheses, available actions, predictions, evidence, and commitments. Module Spaces provide the action-conditioned predictors, memories, reasoners, and controllers that update the Frame. The Attention Metaprotocol chooses which prediction, observation, experiment, or action to perform next. The Frame is not the whole world model; it is the active projection of a larger evidence-backed world-model braid into the current cognitive episode.

PLN contributes a constructive bridge from reasoning to imagination. A proof DAG can be compiled into a procedural generator whose intermediate Atoms become latent variables and whose rule applications become stochastic kernels. The resulting witness-worlds remain marked as hypothetical and retain their inference trails, so they can support planning, explanation, and counterfactual comparison without being mistaken for independent evidence[20].

1.4 The OmegaClaw Control Fabric

OmegaClaw–the current name for the framework called MeTTaClaw in early-2026 design documents–began as a very compact MeTTa agent capable of receiving instructions, reasoning about how to satisfy them, and using tools. Its strategic importance is not the basic tool-use loop, which many agent frameworks provide. Its importance is that the loop itself lives in MeTTa and Atomspace. The agent's working state, its memories, the objects it manipulates, the outputs of cognitive modules, and the rules deciding what to do next can all be represented in the same metagraph substrate.

The extended OmegaClaw architecture introduces three especially important abstractions:

- **Module Spaces** wrap LLMs, open-source transformers, code executors, sensor streams, databases, PLN, MOSES, ECAN, WILLIAM, predictive-coding systems, and remote Hyperon instances behind a uniform capability interface.
- **Context Frames** hold the authoritative state of a multi-step cognitive task: goals, hypotheses, plans, intermediate results, confidence, provenance, resource use, branching alternatives, and completion criteria. The Frame, rather than any one model's context window, is the durable unit of cognition.

- **The Attention Metaprotocol** allocates computation at two timescales. A fast loop handles moment-to-moment importance, activation spreading, and scheduling. A slow loop assesses strategic progress, budgets, stalled hypotheses, and the need to replan or seek human input.

Around these abstractions sit a cognitive task planner, a distributed Atomspace manager, a PRIMUS component manager, cross-node communication, and a self-modification governor. Together they make it possible to deploy a useful LLM-and-tool system now and replace its stand-ins incrementally with deeper PRIMUS mechanisms later. The migration unit is a capability behind a Module Space, not the whole architecture. A component can be benchmarked in parallel, run in shadow mode, promoted when it is ready, retained as a fallback, and rolled back if its real-world behavior degrades[2].

1.5 OmegaSelf: Evidence-Grounded Self-Modeling at the Causal Seam

A Hyperon control fabric needs more than a record of the external world. It also needs a reliable, uncertainty-aware account of the running system itself: which processes and modules are present, which capabilities are reachable and authorized, what the system is committed to, how well it predicts its own actions, and what sequence of changes produced its current state. OmegaSelf supplies this missing layer. It is an event-sourced, PLN-interpreted self-theory and governance system that sits at the causal seam between OmegaClaw's parsed action proposals and their execution[6].

The core stance is deliberately non-anthropomorphic. The self-model is not a privileged ego atom, a polished autobiography, or an LLM-generated personality summary. It is a live, defeasible, context-sensitive theory assembled from mechanically grounded observations: process and workspace probes, permission checks, authenticated messages, module and provider receipts, tests, resource measurements, mutation records, prediction outcomes, and external evaluations. Observations are appended to a durable ledger and projected into Atomspace views. Corrections, retractions, staleness judgments, and context changes are new records rather than destructive rewrites.

OmegaSelf keeps distinct facets of self-knowledge linked but separable. A module may be installed but unreachable, reachable but unauthorized, authorized but ineffective for the current task, or effective under one provider and evaluation rubric but not another. The system therefore treats capability as a contextual inference chain rather than a timeless claim. It likewise separates substrate, epistemic state, commitments, social attribution, continuity, appraisal, and governance. Disagreement among these facets is diagnostic data, not something to smooth into one narrative.

A self-belief becomes operationally useful when it makes a falsifiable prediction. Before a consequential action or probe, OmegaSelf commits an expected outcome, probability, latency or cost where relevant, and an evaluation rubric. After execution it records the receipt, scores the prediction, appends discrepancy evidence, and revises only the applicable contextual belief. Parsed skill calls become immutable typed proposals. A separately authorized policy gate returns one of `Allow`, `Deny`, `RequireProbe`, `RequireReview`, or `Defer`; only an exact proposal carrying `Allow` can reach execution. PLN estimates competence and risk, but does not grant permission.

OmegaSelf also grounds the indexical "this agent, here, now" without pretending that a copied name, memory, ledger, or secret establishes exclusive identity. A short-lived `SelfHereNow` anchor is resolved from authenticated runtime identity, process and workspace boundaries, controlled capabilities, active session bindings, lineage and provenance heads, and at least one renewable proof of current causal control. Restarts, migrations, forks, and merges are represented through directed transports between versioned snapshots. Continuity can branch; rollback creates a new continuation rather than erasing the intervening history.

The smallest useful OmegaSelf implementation answers three questions with named causal consumers: What can I do here? What am I committed to doing? What changed me into my present state? These drive capability-aware dispatch, commitment-conflict deliberation, and a mutation continuity gate. This makes OmegaSelf a practical complement to PRIMUS: PRIMUS supplies deeper models of reasoning, motivation, attention, and learning, while OmegaSelf is designed to make claims about the running agent evidence-bearing and predictive, and to connect those claims to enforceable control flow. Calibration, coverage, and control effectiveness remain measurable properties rather than assumptions.

1.6 OmegaHive: Collective Cognition as an Engineering Laboratory

OmegaHive applies the same principles to a population of agents. The first proposed experiment, OmegaHive1, combines Atomspace-backed OmegaClaw agents with skill-specialist OpenClaw agents in a cooperative system aimed at producing real research, formalization, software, and written artifacts while also studying what makes agent collectives productive rather than merely active[5].

The hive architecture is deliberately more than a group chat. Fast working traffic moves over an internal message bus; a curated human-legible tier carries intentions, decisions, disagreements, escalations, and digests. A shared task board gives every substantial task an owner and status. Shared and individual Atomspaces support both stigmergic coordination and role-specific cognition. Documents require provenance companions that link claims to tasks, messages, and sources. A version-controlled constitution defines communication norms and permission tiers. A reflective conscience role monitors value drift and unhealthy dynamics. Outbound capabilities are enforced by credentials and network routes, not by prompts alone. Deterministic metrics track cost, throughput, blocked work, and escalation, while a human-only recovery path remains available at all times.

OmegaHive should not be confused with a claim that a collection of current agents is already a unified AGI. It is better viewed as three things at once: a practical production system, an externalized cognitive architecture whose coordination mechanisms can be measured, and a test environment for the future distribution of PRIMUS processes across multiple OmegaClaw nodes. The hive can expose failures–message loops, summarization cascades, weak task ownership, cost drift, permission mistakes, or unproductive role specialization–early, while the system is still understandable and humans can intervene.

1.7 A Predictive-Coding and Symbolic-Head Bridge to Open-Source Transformers

Hyperon supports three neural integration modes: existing neural models outside Atomspace, exposed through Spaces; neural computation inside Atomspace, for example through QuantiMORK; and a middle path that can deliver substantial neurosymbolic interaction without waiting for a fully native neural runtime.

In this bridge mode, an open-source transformer is treated as a high-capacity but incomplete cognitive kernel. The initial implementation can freeze most or all of its weights. Symbolic heads connect selected hidden states to Atomspace so that the model can retrieve typed structures, rules, episodic records, proof fragments, task frames, and other mined templates; the heads can also write candidate structures and confidence signals back to Atomspace. A predictive-coding graph sits above or around the relevant transformer layers, using local prediction errors to reconcile bottom-up neural states, top-down task and symbolic expectations, and lateral memory constraints. Sparse rank-one residual columns can provide narrow, auditable corrections, while optional training-free loops over a stable middle transformer window can expose additional latent computation at inference time.

This is not one fixed architecture. FabricPC is one practical scaffold because it supports arbitrary feedforward, recurrent, skip, and cyclic graphs, heterogeneous nodes including transformer blocks and associative memories, local predictive-coding dynamics, JAX acceleration, and controlled comparisons between predictive coding and backpropagation on the same topology[8]. Looped columnar residual adaptation is one candidate for combining extra inference-time computation with sparse, weakness-guided, causally auditable residual modules[12]. Other implementations can preserve the same architectural role.

The key point is that the bridge is controlled by the rest of Hyperon. OmegaClaw records its state in Context Frames, invokes it as one or more Module Spaces, chooses compute budgets and loop policies, routes symbolic hypotheses to PLN or WILLIAM, and records causal audits and performance traces. PRIMUS supplies the longer-term cognitive logic: what should be predicted, which discrepancies matter, when a symbolic structure should be promoted, and how a learned specialization should be composed with motives, options, and reasoning.

1.8 Major Architectural Extensions

The principal extensions covered here are:

- the five-layer public framing of Hyperon as MeTTa, knowledge representations, AI algorithms, PRIMUS, and ASI:Chain;
- OmegaClaw as a concrete control fabric that spans those layers and supports a measured migration from LLM-centered systems toward PRIMUS-centered systems;
- OmegaSelf as an event-sourced, predictive, faceted self-theory and externally rooted governance layer at the proposal-to-execution seam;
- selective lessons from the ClarityOmega design concerning full-intent mutation gates, conversation-scoped aliveness, agency provenance, evidence-grounded capability, and the need for every detector to have a causal consumer;
- OmegaHive as a governed, observable, provenance-aware experimental architecture for cooperative agent fleets with per-agent and collective self-modeling;
- a three-mode neural integration story–external, bridge, and native–with the bridge centered on open-source transformers, symbolic heads, predictive coding, sparse residual adaptation, and optional looped computation;
- a sharper separation between cognitive architecture, operational orchestration, and verifiable runtime infrastructure;
- world modeling as an action-first, evidence-anchored, multi-rate braid across symbolic Atoms, HMH structured associative memory, and dense neural dynamics, with OmegaClaw Context Frames serving as active operational projections of this world model;
- the progression from πPLN to ωPLN: persistent context-indexed positive and negative evidence, temporary local Bayesian charts, provenance-aware revision, reversible residuals, and separate diagnostics for evidence multiplicity, path holonomy, and chart non-gluability;
- TECAN as a typed, cost-accounted refinement of ECAN, together with budgeted STLM pattern mining, dependency towers, semantic algorithmic chemistry, and procedural world-generators compiled from PLN proof DAGs;

- an expanded experimental agenda emphasizing ablation, calibration, continual learning, causal audits, task-level productivity, governance, and rollback.

1.9 Status and Scope

Hyperon is simultaneously a body of working open-source software, a set of concrete engineering programmes, and a research programme containing hypotheses that still need to be tested at scale. This whitepaper intentionally discusses all three. A statement that a component "can" support a function may therefore refer to an available implementation, a specified near-term design, or a mathematically motivated research direction. Where maturity matters, the text distinguishes these cases explicitly. The architecture should be judged not by treating every proposed component as already complete, but by whether the interfaces, experiments, and migration paths make progress cumulative and falsifiable.

To make that distinction operational, this whitepaper uses the following status vocabulary.

- **Implemented capability**: working code or a running service exists for the stated scope; this label does not by itself imply production hardening or external validation.
- **Prototype**: an integrated or partially integrated implementation exists, but interfaces, coverage, reliability, or scale are still under active development.
- **Specified design**: the interfaces and intended behavior are described in enough detail to guide implementation, while complete integration is not claimed.
- **Research hypothesis**: a mathematically or cognitively motivated proposal whose value must be established by proof, experiment, ablation, or all three.
- **Illustrative target**: a numerical objective, acceptance criterion, or scenario parameter used to make an engineering plan concrete; it is not an observed result unless a versioned benchmark artifact is cited.
- **Externally validated**: a result reproduced or reviewed by an independent party under disclosed conditions. No statement receives this label merely because it is plausible, implemented, or internally measured.

The table below is the baseline status register for major subsystems. More specific labels in the body take precedence. Throughout the document, claim status is determined by the stated evidence boundary: implemented capability, engineering prototype, specified design, illustrative target, research hypothesis, or externally validated result.

Table 1: Baseline maturity status for major Hyperon subsystems and claim classes.

Subsystem or claim class	**Baseline status**	**Scope of the status**
MeTTa, Atomspace, and MORK core	Implemented capability	Working open-source components; exact performance and operational limits remain version- and workload-dependent.
DAS, RSpace/Rholang, and ASI:Chain integration	Implemented capability / prototype	Core technologies exist; the end-to-end Hyperon deployment profile varies by integration and governance configuration.

Subsystem or claim class	Baseline status	Scope of the status
MeTTa-IL, PeTTa, JeTTa, PyMeTTa	Implemented capability / prototype	Different compiler and developer paths have different coverage and maturity; comparative performance requires versioned benchmarks.
PRIMUS algorithms and shared cognitive architecture	Implemented components / specified design	Several algorithms have working implementations; the complete continuously integrated architecture remains an engineering programme.
OmegaClaw	Prototype / active implementation	Operational control-fabric functions are being implemented and exercised incrementally.
OmegaSelf	Specified design / prototype components	Evidence, prediction, policy, continuity, and governance mechanisms are specified; full deployment and validation remain staged.
OmegaHive	Experimental prototype	A research and engineering environment for governed multi-agent cooperation, not a claim of solved collective intelligence.
Transformer bridge and QuantiMORK directions	Prototype / research hypothesis	External integration is practical now; deeper predictive-coding, symbolic-head, and native neural integration require staged experiments and ablations.
ωPLN, TECAN, STLM, semantic chemistry	Specified design / research programme	Formal and algorithmic proposals with selected implementation paths; broad performance claims require benchmark evidence.
Weakness, geodesic control, TransWeave, SubRep, MetaMo, and fluid formulations	Research hypothesis / specified design	Strong organizing ideas with conditional formal statements; engineering benefits and cross-domain generality require proof obligations and empirical validation named in the relevant sections.
Capacity, latency, efficiency, and rollout numbers	Preliminary observation / illustrative target	Not externally validated in this whitepaper unless accompanied by a versioned benchmark package.
Beneficial AGI and AGI-to-ASI pathway	Platform thesis / research objective	Hyperon provides unusually explicit handles for integration, measurement, governance, and correction; beneficial outcomes remain objectives to be tested rather than consequences of architecture alone.

Throughout this paper, the words "certificate," "guarantee," and "proof" are used in their strict sense only when the artifact's scope, assumptions, theorem, or test coverage are stated. Otherwise, a "certificate" is a scoped machine-readable validation or admission report, and its conclusions do not automatically extend out of distribution, across versions, or to whole-system safety.

1.10 From AGI toward Beneficial ASI

Hyperon and PRIMUS are designed not only to assemble broad competence, but to make growth itself inspectable. Goals and motives are explicit objects. Reasoning carries provenance and uncer-

tainty. Attention and budgets can be audited. Self-modifications can be proposed as typed graph transformations, evaluated in shadow or twin environments, protected by invariants, and rolled back. OmegaClaw adds an operational governor around this process. OmegaSelf adds immutable evidence, pre-action prediction, renewable runtime identity, directed continuity, and a separately authorized decision gate. OmegaHive adds social and organizational checks such as separated roles, human-visible escalation, permission tiers, and independent quantitative and qualitative evaluation. ASI:Chain can add persistent, cross-organization verification where that level of trust is warranted.

No architecture alone guarantees beneficial superintelligence. The claim here is more specific and more testable: a system whose knowledge, motives, control rules, module choices, edits, and provenance share a transparent substrate offers better handles for measurement, governance, and correction than a system in which these functions remain implicit in one opaque parameter array. Hyperon's path from useful systems to AGI and beyond is therefore intended to be incremental, pluralistic, reversible, and open to continuous empirical challenge.

2 Hyperon System Design

Modern commercial AI systems are often complex multi-component hybrids. For example, LLM-based assistants commonly combine transformer networks with retrieval-augmented generation, knowledge graphs, domain-specific search, formal-reasoning verifiers, and other tools. Many such systems retain a fundamental architectural limitation: they are collections of separate components that cannot exchange sufficiently rich semantic state. A deployment may combine a knowledge store, neural-network trainer, reasoning engine, and orchestration layer, yet sharing only queries, results, and compressed context does not enable the components to cooperate as fully as they could. Richer intermediate cognitive-processing states must cross the boundaries. Otherwise data may be copied repeatedly, caches may diverge, and interfaces may become brittle semantic bottlenecks because the components use different representational languages with only shallow translation.

Hyperon takes a structurally different approach. Rather than restricting components to endpoint APIs, facts, rules, neural embeddings, activations, goals, and even schedulers are represented as first-class objects in a shared Atomspace. Cognitive modules can therefore inspect and manipulate common typed structures, enabling interactions that would otherwise require complex, custom integration across multiple representational boundaries.

2.1 The Technical Foundation

The Atomspace is implemented as a typed, content-addressed metagraph in which each represented item is an "Atom." Atoms can range from simple facts to complex procedures, neural-network weights, and control signals. The architectural value lies in their shared representation, indexing, and access mechanisms.

Behind the Atomspace are several back ends with different properties. The discussion focuses on MORK, an engine designed for high-performance Atomspace operation and already usable for the scoped functions described here.

Claim status: Implemented capability; performance characterization is preliminary and workload-scoped.

MORK provides a Merkle-DAG/PathMap trie structure, lock-free update paths for concurrent access, and memory layouts intended to support both symbolic operations and dense numeric computations. For bounded-depth, cache-resident paths under favorable schemas and query mixes, lookup latency can behave close to constant over the tested working set. This is a scoped engineering observation,

not a workload-independent asymptotic complexity theorem. Cost depends on Atom encoding, trie depth and branching, index design, memory residency, cache state, query selectivity, update rate, contention, concurrency, storage tier, and software version; adversarial or cache-cold workloads can behave differently. Future developments may augment MORK with alternative or superior Atomspace backends. MORK is working now and has shown favorable preliminary behavior in internal observations for selected algorithms and datasets. Reproducible performance claims are to be reported with the data model, query mix, hardware, memory assumptions, cache state, concurrency, software version, warm-up procedure, latency distributions, failure rates, comparison baselines, acceptance thresholds, and raw results needed for independent reproduction.

The MeTTa programming language is a compact, abstract scheme for human or automated construction of programs as coupled metagraph rewrite rules. MeTTa programs are themselves represented in Atomspace, enabling concise implementation of complex cognitive algorithms and system-level reflection. This self-referential design provides concrete machinery for self-modeling and governed self-modification, whose contribution to beneficial AGI is evaluated through the controls and evidence obligations developed below.

The system scales through DAS (Distributed Atomspace), which extends the same unified space across clusters using MongoDB/Redis backends. Meanwhile, RSpace/Rholang integration – via a "MeTTaCycle" infrastructure that compiles MeTTa into a combination of MORK and rholang operations – enables capability-secured execution for multi-tenant scenarios and blockchain-based commitment, ordering, and linkage for provenance records. Semantic validity and authorship require the separate controls described below.

2.2 The Space API: A Universal Interface

The Atomspace is Hyperon's uniform cognitive substrate–analogous to shared digital brain tissue–where different kinds of data and processing can coexist and interact. As a brain has specialized regions, Hyperon has "Spaces": specialized environments that connect to Atomspace through a common declared interface.

A Space may be an in-RAM knowledge graph adaptively updated by reasoning and learning algorithms, a distributed database shard, a connection to a neural-network service, or a blockchain-based smart-contract executor. The architectural advantage is that MeTTa code is substantially Space-independent: pattern matching, rule firing, and data manipulation use the same declared interface across supported implementations. Different Spaces may use different back ends–for example MORK, JeTTa under development for JVM and Android deployment, or a floating-point-centric representation specialized for neural networks.

2.2.1 How Spaces Work

The Space API defines a standard interface that a conforming backend can implement. The core operations–match, bind, and rewrite–retain the same declared semantics whether the caller accesses local RAM, queries a distributed database, or invokes a neural network; latency, consistency, availability, and failure behavior remain backend-specific operational properties.

Examples currently implemented or under active investigation include:

- MORK backed Atomspace for scalable in-RAM cognitive processing
- DAS for distributed storage and large-scale queries
- Rholang Atomspace for concurrent, capability-secured execution

- Neural Spaces that make deep neural networks appear as queryable Atomspaces

The Space abstraction layer lets cognitive processes use different computational substrates through a common interface, reducing special-case integration code while retaining explicit backend-specific constraints.

2.3 MORK: The High-Performance Core

MORK is a primary storage and execution engine for Hyperon cognition. It stores the Atomspace metagraph in a content-addressed trie structure optimized for pattern matching, unification, versioning, and diverse cognitive operations. Whether a release meets real-time, capacity, or concurrency requirements is determined by workload-scoped benchmark records that identify the software version, hardware, Atom schema, index design, cache state, query mix, concurrency, warm-up procedure, latency distributions, failure rates, baselines, acceptance thresholds, and raw results.

The key innovation is the PathMap/Merkle-DAG structure. Every Atom is placed at a defined location in a prefix-compressed trie. Nodes are immutable, so updates create new versions rather than modifying existing nodes. Writers prepare compact "deltas" that are merged atomically. Under the backend's stated snapshot and merge semantics, readers observe a consistent version while multiple writers are active. The data structure is optimized for pattern-matching and unification queries that lie at the core of many AGI-oriented algorithms; measured performance remains workload- and version-dependent.

This architecture addresses a critical problem: allowing many cognitive processes–inference, learning, pattern mining, and attention–to operate concurrently on a shared large-scale knowledge graph while limiting synchronization bottlenecks under declared merge and conflict semantics.

2.3.1 Weighted Atom Sweeps: Cognitive Scheduling

Along with pattern matching and unification queries, MORK supports alternate access modes such as Weighted Atom Sweeps (WAS), which combines priority-weighted traversal with attention-management functions for a cognitive system.

Each node in the PathMap carries an aggregate weight derived from its children. This creates a probability tree that supports weighted sampling of Atoms according to a declared importance measure, such as recency, relevance, or uncertainty. Sampling cost and quality depend on tree shape, update policy, cache state, and workload and are therefore measured rather than assumed.

The system provides two main traversal strategies:

1. **Weighted random walks** that descend the tree by probabilistically selecting children based on their weights
2. **Chunked priority queues** that maintain sorted lists of high-priority regions for systematic processing

Where the selected operations are lock-free and their merge semantics are explicitly commutative, multiple sweep processes can run in parallel. This acts like an operating system scheduler for cognition–background inference can trickle through less-important regions while urgent pattern recognition focuses on hot spots, with coordination handled by the data structure and declared conflict rules rather than by one global application lock.

2.3.2 Executing Dense Kernels: ShardZipper and ByteFlow

MORK's standard compute API is designed primarily for sparse kernels such as pattern matching and unification. Dense-kernel support is an active engineering programme rather than a completed performance claim. Two complementary approaches have been specified:

ShardZipper is designed for deterministic batch processing. When a workflow must extract a large portion of the graph, run intensive computation on it–for example on a GPU–and merge the result back, ShardZipper supplies the required structure. It defines shards through hash prefixes and maintains a "one-hole context" (technically, a McBride derivative) that permits a shard to be extracted, processed in isolation, and reinserted with Merkle-root linkage and integrity checks under the stated tree construction.

ShardZipper can be implemented against current MORK interfaces. ByteFlow takes a different approach for scenarios in which hot regions drift over time and requires deeper changes to the MORK infrastructure. Once implemented and benchmarked, it may outperform ShardZipper for selected dynamic workloads. The algorithm continuously repacks frequently accessed subtrees into contiguous parent-pointer blocks (PPTNs) that can be supplied directly to kernels without pointer chasing. A routing table maps key prefixes to these packed blocks, and the system splits or merges blocks as access patterns change without requiring full re-serialization.

The current design favors ShardZipper when:

- Computation is concentrated in stable, well-defined regions
- Deterministic, reproducible batch processing is required
- Processing must be distributed across multiple devices with explicit boundaries

ByteFlow is favored when:

- Access patterns change dynamically based on cognitive focus
- Low kernel-operation latency is the primary objective
- The layout should adapt to changing workload patterns

Implementation and comparative testing will determine the workload boundary between these dense-kernel approaches.

2.4 Neural Network Integration

Hyperon's approach to neural networks now has three complementary modes rather than a single all-or-nothing choice. The modes differ in how deeply neural state is represented inside Hyperon, but all of them expose typed results, provenance, confidence, and control signals through Spaces and Atomspaces.

External integration through Neural Spaces is the pragmatic route for existing models. Large language models, vision networks, speech systems, and specialized predictors remain in their native frameworks and are invoked as services. Their outputs, embeddings, selected activations, uncertainty estimates, and audit traces are represented as Atoms. MeTTa code can attach truth values, compare models, route outputs to PLN, record them in Context Frames, and treat the neural system as one capability among many. This mode is appropriate when model internals are inaccessible, when operational stability matters more than deep integration, or when a commercial or remote endpoint is temporarily the best available engine.

Bridge integration through predictive coding and symbolic heads is the new middle path. It assumes an open-source transformer or other instrumentable neural network whose hidden states and selected layers can be observed and, if useful, lightly adapted. Symbolic heads connect those states to Atomspace memories and rules. Predictive-coding nodes reconcile bottom-up hidden states with top-down symbolic, contextual, and task predictions. Sparse residual columns can make local, causally auditable corrections while most base weights remain frozen. Optional controlled looping of a stable middle layer window can allocate extra inference-time computation. The bridge therefore creates a structured "cognitive cortex" around an existing neural kernel without requiring the kernel to be reimplemented inside Atomspace.

Native integration through QuantiMORK and related Neural Spaces is the most radical route. Neural tensors and multiresolution structures are represented directly in a MORK-backed Atomspace, allowing attention, convolution, prediction-error updates, pattern mining, and symbolic rewrites to operate over shared structures and caches. Wavelet- or hierarchy-oriented neural architectures are especially attractive because their multiscale organization can map naturally onto MORK's prefix-tree structures.

These three modes are intended to coexist. A single OmegaClaw-controlled system may use a remote frontier model for natural-language generation, a local open-source transformer with predictive-coding and symbolic-head overlays for task cognition, and a native QuantiMORK network for selected high-bandwidth perception or continual learning. The Module Space registry presents a common capability interface, while Context Frames and the Attention Metaprotocol decide which mode is appropriate for each subtask.

2.4.1 When to Use Each Mode

External integration minimizes engineering risk and maximizes immediate model choice. Bridge integration is attractive when open weights and activations are available and the goal is to add structured memory, local adaptation, continual learning, or causal audit without retraining a large model end to end. Native integration is most compelling when symbolic and neural operations must share fine-grained state, when computation should be sparse and locally scheduled, or when the neural architecture itself is being redesigned for Hyperon.

The detailed bridge architecture, including FabricPC, symbolic heads, sparse columnar residuals, and looped inference, is developed in Section 10.

2.5 Distributed Execution and Scaling

Efficient single-machine memory use remains important, and MORK is designed to extend that capacity. Real-world AGI systems will nevertheless need to scale beyond one machine. DAS (Distributed Atomspace) extends Hyperon's cognitive space across clusters while presenting a common Atomspace interface rather than claiming identical local and distributed behavior.

DAS uses MongoDB and Redis to shard the metagraph and maintains indexes for traversal, property queries, and pattern matching. MeTTa code uses the same query interface for local and distributed Atoms, while routing, latency, consistency, availability, partition behavior, and failure recovery remain explicit distributed-system properties.

2.5.1 Decentralized Execution

For scenarios requiring multi-party collaboration or high-security execution, Hyperon has a "MeTTa-Cycle" layer that supports running MeTTa code as Rholang processes on RSpace. This provides:

- **Capability-based security**: Fine-grained control over what code can access.
- **Commitment integrity and linkage**: Hashes, signatures, transaction records, and Merkle inclusion proofs can show that specified bytes or commitments were recorded and linked relative to a named root and consensus history.
- **Distributed consensus**: Multiple parties can contribute to cognitive processes under an explicit consensus and authorization model.

This is not just about scaling; it enables new models of collaborative AI development in which different organizations can contribute cognitive modules while maintaining scoped security and attribution. A chain record does not, by itself, prove that the committed representation is true, semantically correct, causally valid, beneficial, authorized, or continuously available. Authorship requires signatures and an identity or key-governance process; availability requires replication, pinning, or an availability service; semantic and execution validity require independent validators, replay, tests, or formal verification; authorization requires policy; and disputes require an explicit resolution and correction process. These properties are deliberately treated as distinct: content hashes and chain commitments support integrity and linkage; signatures and key governance support attributed authorship; replication and pinning support availability; validators, replay, tests, and formal methods support semantic or execution assessment; policy supports authorization; and governance procedures support dispute resolution. None of these mechanisms alone establishes all of the others.

2.6 State Management and Persistence

A thinking system needs to manage state carefully–not just for persistence, but for reproducibility, debugging, and collaborative development. Hyperon's State Management Subsystem (SMS) provides a thin layer that treats engine internals as opaque while offering essential services.

The system captures state changes as AtomspaceUpdate patches that bundle link insertions and weight modifications from each cognitive epoch. Four key artifacts get stored:

- **CheckpointRef**: Snapshot identifiers for full state recovery
- **UpdatePatch**: Incremental changes since the last checkpoint
- **ChangeDigest**: Summaries for quick difference detection
- **InfluenceSketch**: Approximate impact assessments for optimization

For edge weights that may increase or decrease, SMS uses a PN-counter or a domain-specific bounded counter rather than a grow-only G-counter[24]. For replicas $r \in R$, a PN-counter stores two grow-only component maps $P, N : R \to \mathbb{N}$, with value and merge

$$w(P,N) = \sum_{r\in R} P(r) - \sum_{r\in R} N(r), \qquad (P,N) \sqcup (P',N') = (\max(P,P'), \max(N,N')), \tag{1}$$

where maxima are componentwise. Increments update the local P component and decrements update the local N component, so concurrent operations converge without treating a decrease as an increment. A transfer is represented as an idempotent, transaction-identified debit/credit pair, not as one counter

mutation; if a resource must never go negative, an escrow or bounded-counter CRDT allocates spendable rights before the debit. Policy-invalid, duplicated, or unmatched transfer records are quarantined or compensated under a deterministic reconciliation rule. The invariants are replica convergence, idempotent replay, preservation of transaction identity, and–where required–a nonnegative domain balance. Rholang lens contracts provide capability-controlled access to the specific query patterns, traversal depths, and mutation classes.

2.7 Putting It All Together

The following trace shows how these components cooperate during a representative neurosymbolic operation:

1. **Selection**: A WAS (Weighted Atom Sweep) process identifies a hot region of the graph that needs attention, perhaps because new information has arrived or uncertainty is high.
2. **Layout**: ByteFlow's routing table maps this region to a parent-pointer block containing the relevant Atoms in a kernel-friendly format.
3. **Computation**: A GPU kernel processes the block–perhaps running attention mechanisms or updating neural weights–and produces a compact patch describing what changed.
4. **Integration**: The host system replays this patch into the PPTN. If the changes exceed certain thresholds, ByteFlow might split or merge blocks to maintain optimal layouts.
5. **Propagation**: MeTTa rules detect the changes and trigger appropriate responses–perhaps PLN factor-graph messages need to propagate new evidence, or pattern miners should look for newly-emergent structures.

In the native Atomspace mode, this process can stay in one address space, use shared indices, and maintain common invariants. The external and bridge modes retain some runtime boundaries, but OmegaClaw makes those boundaries typed, observable, and progressively replaceable rather than leaving them as ad-hoc text interfaces.

3 The Language Stack

3.1 A Multi-Layer Approach to Cognitive Programming

Building cognitive systems requires balancing competing demands, on the developer-interfacing level as well as the underlying cognitive-dynamics level. Developers need a language that is intuitive enough to express complex cognitive patterns easily. The system needs a compact, unambiguous representation for compilation and distribution. The runtime needs low-level operations that can manipulate data structures efficiently. And increasingly, teams want to leverage Python's ecosystem and LLM-assisted development.

Hyperon addresses these needs through a carefully designed stack of languages, each serving a specific purpose while maintaining semantic consistency:

- **MeTTa**: The high-level language where developers write abstract cognitive code
- **MeTTa-IL**: The intermediate representation used for compilation and optimization
- **MORKL/MM2**: Low-level languages that run directly on the graph structures

- **PyMeTTa**: A Python-compatible layer for rapid prototyping and integration

This multiplicity of language tools is not reinvention; it provides a transformation and execution path across abstraction levels while preserving the declared requirements of cognitive computation.

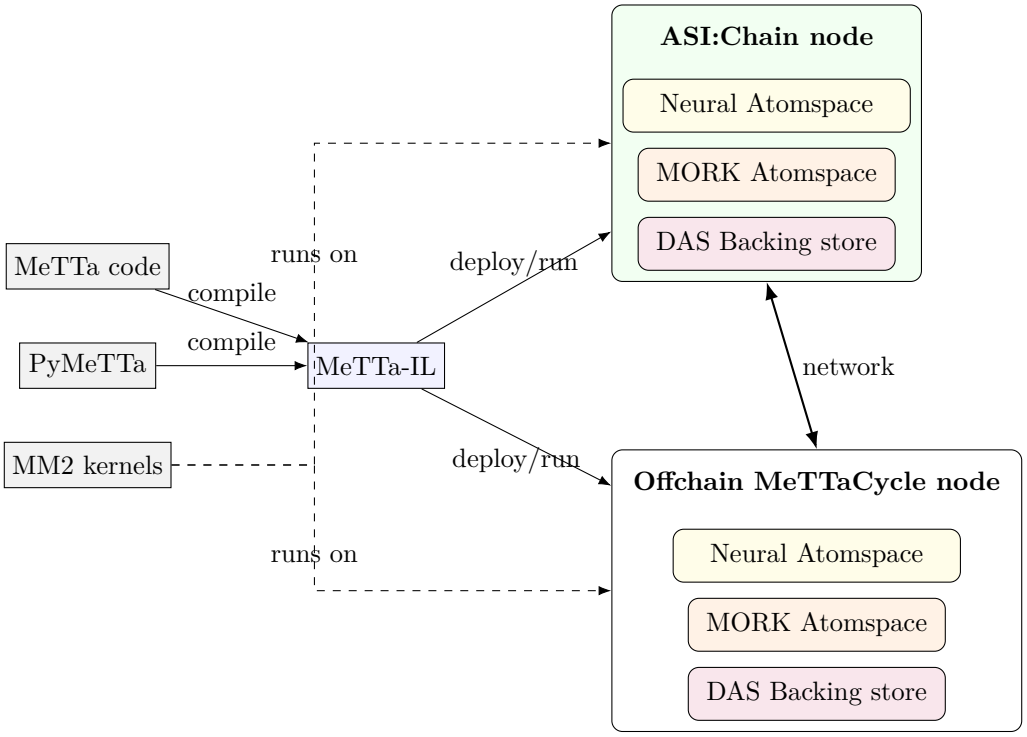

Figure 2: High-level software architecture: MeTTa/PyMeTTa compile to MeTTa-IL which leverages MORK Atomspace; MM2 runs on MORK Atomspace; both ASI:Chain and Offchain MeTTaCycle nodes contain full Atomspace stacks with DAS backing. Nodes may in general contain numerous Atomspaces not just 3.

3.2 The Technical Stack

Each layer in the stack has precise responsibilities:

MeTTa provides a declarative, homoiconic pattern-rewrite language built specifically for Atomspace manipulation. It operates at a higher level of abstraction than other functional or logic programming languages, allowing flexible experimentation (by humans or AI modules) with a wide variety of type systems, language constructs, logics and so forth – while still retaining speed and scalability in a wide variety of situations. It features committed-choice evaluation, explicit Space management, and built-in support for nondeterministic streams.

MeTTa-IL serves as the compiler-friendly intermediate representation derived from a Graph-Structured Lambda Theory (GSLT) description. This is what actually gets distributed, optimized, and compiled to various backends including MORK, Rholang, JAX, and Rust kernels.

MORKL/MM2 implements kernel-level typed operations that work directly with MORK's data structures–factor-graph message passing, weighted sweeps, and ByteFlow operations for GPU/TPU acceleration. Highly performance sensitive algorithms that need to operate across MORK at scale may be most effectively implemented in MM2.

PyMeTTa and the metta-magic library (still under development, expected to launch in H2 2026) provide a restricted Python dialect that transpiles cleanly to MeTTa-IL, plus a comprehensive library suite. This enables notebook-based development, seamless LLM assistance, and integration with the broader Python ecosystem.

3.3 MeTTa: Cognitive Code as Graph Transformations

MeTTa takes a distinctive approach to programming: developers write small equations, queries, and control patterns that match against subgraphs and rewrite them. The language keeps results as lazy nondeterministic streams until an explicit collapse operation is applied, supporting cognitive processes in which multiple possibilities are explored in parallel.

MeTTa code can target named Spaces, including a local Atomspace, distributed DAS shards, or neural Spaces. The same declared pattern-matching and rewrite interface applies from small knowledge bases to cluster-scale deployments, while backend-specific performance, consistency, availability, and resource behavior remain explicit.

3.3.1 The Execution Model

MeTTa-4 is a semantic model for MeTTa, intended to provide a clean, small-step operational semantics that aligns directly with the surface syntax. The core includes:

- **Committed choice by default**: Once a branch is selected, it proceeds deterministically
- **Delimited nondeterminism**: Explicit operators for forking and rejoining computation streams
- **Atomic Space operations**: The for construct provides transactional semantics across Spaces

The specification offers both an abstract machine view and an equivalent MeTTa-calculus formulation. The compiler pipeline cleanly separates kernel operations (rewrite/match/space ops), macro expansions (let/if constructs), and environment management (modules/tokens), enabling efficient compilation to the intermediate representation.

3.4 MeTTa-IL: The Compilation Bridge

MeTTa-IL (IL = Intermediate Language) is intended to make program semantics explicit when code crosses system boundaries. Each supported construct is represented explicitly and typed in the IL, reducing ambiguity about evaluation order, Space selection, and type conversion within the specified language fragment.

This intermediate form enables several critical capabilities:

- Generate both interpreters and type systems from the same specification
- Target multiple execution backends without semantic drift
- Apply optimizations only after equivalence, regression, and type checks establish the declared behavior within the tested scope
- Verify named conformance, integrity, or safety properties under stated assumptions before deployment

3.4.1 The GSLT Foundation

The system uses Graph-Structured Lambda Theory as a formal foundation. For a supported domain–including normalization, satisfiability checking, or evolutionary computation–a GSLT specification captures the intended structure. The toolchain can then derive an interpreter and a corresponding type system; the resulting semantic-preservation claim is conditional on the stated GSLT, generator, and validation checks.

MeTTa-IL serves as the concrete syntax consumed by these generators. The design maintains two GSLTs: a fine-grained one for implementation details and optimizations, and a coarse-grained one for developer-facing types. A release claiming correspondence between them must provide the typed maps, assumptions, theorem statement, proof artifact, and replayable environment; otherwise correspondence remains an explicit formalization obligation.

3.5 PeTTa and JeTTa

MeTTa-IL provides a formalization-oriented approach to MeTTa compilation, including a framework for stating and proving semantic-preservation and selected safety properties under explicit models. This is attractive for smart contracts and AGI control paths where mechanically checkable properties matter. Incorporating the rho calculus supplies a defined basis for concurrent behavior; robustness across workloads and runtimes still requires theorem-specific analysis and empirical testing. Alternative MeTTa compilers may be more convenient when these properties are not central.

JeTTa is a framework for compiling MeTTa to the JVM using Kotlin infrastructure. This permits use of JVM garbage collection and supports deployment of MeTTa code on Android devices, subject to the feature coverage and conformance status of the relevant release.

PeTTa is a MeTTa compiler developed in SWI-Prolog using recursion schemas. Preliminary developer experience indicates execution performance in the range of modern Prolog systems and competitive with a number of functional-programming runtimes for selected workloads; a comparative claim is meaningful only with named versions, programs, hardware, warm-up, and measurement procedures.

PeTTa can also connect to MORK as a back end to increase capacity and scalability.

Claim status: Implemented/prototype integration; the capacity figures below are preliminary engineering observations, not reproducible or externally validated benchmark results.

At the time of the original observations, developers reported approximately 50–100 million Atoms resident in RAM for PeTTa alone on a powerful development machine, and approximately 500 million Atoms for a PeTTa–MORK integration. These figures are retained because they communicate the engineering scale being pursued, but this whitepaper does not contain the hardware model, software commit, Atom schema, bytes per Atom, index layout, cache policy, query mix, concurrency, warm-up, failure criteria, or raw measurements required to reproduce them. Until a versioned benchmark package supplies those details, they should be read as capacity observations and roadmap evidence, not as guaranteed limits or production sizing guidance. Future capacity and concurrency results should disclose the software commit, hardware and memory configuration, Atom schema, bytes per Atom, index layout, cache policy and state, query mix, concurrency, warm-up procedure, latency and throughput distributions, failure criteria, comparison baselines, acceptance thresholds, and raw measurements.

PeTTa includes substantial concurrency-management functionality and does not itself attempt blockchain integration; MORK handles additional concurrent processing in pattern matching and unification. Future experiments will determine the workloads for which MeTTa-IL provides a practical

performance advantage over PeTTa, separately from MeTTa-IL's role in enabling property analysis. One working hypothesis is that MeTTa-IL will be especially competitive when a workload contains a large fraction of complex symbolic processing outside the MORK operations. That hypothesis is to be tested rather than treated as a settled comparative result.

3.6 MORKL/MM2: Where Hot Loops Live

High-level languages are effective for both human and AI programmers. Some operations in an AI system architecture nevertheless need to run close to the data, with controlled overhead and parallel execution. Factor-graph message passing for probabilistic inference, ECAN attention sweeps, pattern-mining iterations, and kernelized tensor operations are examples, and they form key parts of the computational backbone of cognition in the PRIMUS framework.

While we could implement these in MeTTa, the resulting overhead would be poorly matched to real-time cognitive workloads. MORKL and MM2 provide low-level primitives designed for database-engine-class performance while preserving the declared semantic invariants and exact rewrite behavior required by the logic. Exact speed and scale remain versioned benchmark questions.

MORKL provides a database-query layer for MORK, roughly analogous to SQL for its scoped operations, whereas MM2 supports dynamic algorithms implemented directly against MORK data structures.

3.6.1 Core Operations

The MM2 layer implements multiple critical PRIMUS subsystems; as illustrative examples:

Factor-graph PLN encodes logical variables and rules as Atoms in the graph, with local potentials as typed functions mapping neighboring truth values through a product quantale structure. Messages traverse MORK using PathMap neighbor lookups, while the orchestration layer manages convergence, termination, and independence-error mitigation.

Weighted Atom Sweeps leverage the aggregate weight counters in each trie node to sample subgraphs proportionally to importance, age, or other metrics. The transform functions, written in MeTTa or PyMeTTa, apply local updates that propagate through zipper merges.

3.7 PyMeTTa: Developer Ergonomics

While many AGI researchers embrace high level and flexible languages such as LISP, Prolog or MeTTa, many AI application engineers prefer working in familiar imperative-language-based environments with robust tooling, visualization capabilities, and AI-assisted coding. PyMeTTa (in the early stages of development at the time of writing) provides a Python-compatible surface that transpiles to MeTTa, enabling notebook-driven development while preserving the declared semantics of the core system within the supported transpilation subset.

The metta-magic library brings batteries-included functionality, e.g. covering :

- PLN inference operations
- Evolutionary algorithms (EDA, genetic programming)
- Pattern mining pipelines
- Algorithmic chemistry simulations

- MetaMo control interfaces
- DAS integration utilities
- *and more*

3.7.1 Clean Transpilation

PyMeTTa deliberately restricts Python to a subset that can be unambiguously converted to MeTTa-IL. This means explicit loop and conditional structures, typed decorators for function signatures, and clear module boundaries.

The benefit is substantial: code written in PyMeTTa compiles to the same MeTTa-IL that would be generated from native MeTTa, which can then run on MORK, Rholang, or any other backend without requiring Python in the runtime. This preserves determinism and enables verification while providing Python's development convenience.

3.7.2 Performance Integration

The metta-magic backend registry enables performance-critical routines to be compiled to native Rust or JAX/XLA kernels. The intended execution plan crosses the Python–native boundary once per cognitive step. A 5–50 ms step-duration envelope is an illustrative engineering target for selected interactive workloads, not a measured latency distribution established by this whitepaper. The design objective is for orchestration overhead to remain a small, reported fraction of the total step budget; "negligible" is not used as a substitute for measurement. A reproducible result must disclose payload size, workload and query mix, hardware, software versions, compilation and warm-up, cache state, process and network boundaries, topology, concurrency, comparison baseline, and failure criteria, and must report p50, p95, p99, tail outliers, throughput, and failure rates. Until such a package is published, these numbers are targets for system engineering and acceptance testing.

The library provides comprehensive coverage across algorithms (genetic programming, EDA, PLN, mining, algorithmic chemistry), control systems (MetaMo, ECAN, schedulers), external integrations (DAS, avatars), and contract systems (Rholang compilation, on-chain proof checking).

3.7.3 LLM-Assisted Development

The system treats AI-assisted software development as a first-class engineering workflow. Because PyMeTTa presents a Python-compatible surface, contemporary coding models can support rapid prototyping, test generation, refactoring, and documentation under ordinary review, type checking, and regression-test controls. Modern language models can also generate MeTTa or MM2 in selected cases, but generated code is admitted only through the same compiler, test, benchmark, provenance, and human-review processes as human-authored code.

3.8 Distribution and State Management: Handling Multi-Node Cognition

Real cognitive systems involve multiple machines, changing knowledge graphs, and teams of developers. The Hyperon State Management Subsystem provides essential services while treating engine internals as opaque, ensuring clean abstraction boundaries.

State changes are captured as AtomspaceUpdate macro-patches that bundle link insertions and weight modifications from each inference epoch. The system maintains four key artifacts per batch:

- **CheckpointRef**: Points to complete state snapshots
- **UpdatePatch**: Contains incremental changes since the checkpoint
- **ChangeDigest**: Provides quick difference summaries
- **InfluenceSketch**: Estimates impact for optimization decisions

Updates use the PN-counter, bounded-counter, or transaction-record CRDT selected for the edge's domain semantics, following Equation 1; a grow-only counter is used only for quantities that are genuinely monotone. This preserves convergent replica merges while supporting explicit decreases, decay, and auditable transfers. Rholang lens contracts provide capability-controlled access, restricting queries and mutations to specific templates, traversal depths, and policy-authorized counter operations.

For LLM and RAG workspaces, the same contract system handles vector and graph events. For core Hyperon operations, it can bound cold-replay work and enforce traversal and capability restrictions intended to block data exfiltration through crafted second-order patterns within the stated threat model; adversarial testing and implementation review remain required.

3.9 Sketch of the Development Workflow

An example development flow utilizing these tools might leverage the entire stack:

1. High-level logic gets written in MeTTa (or PyMeTTa for Python users), keeping streams lazy and collapsing only at measurement boundaries
2. Compilation produces MeTTa-IL, the distributable artifact that preserves semantics across backends
3. Hot loops call into MORKL/MM2 kernels that operate directly on ByteFlow blocks or ShardZipper shards
4. Background processing runs via Weighted Atom Sweeps, maintaining attention and performing long-range inference
5. Distribution can target a combination of MORK for performance and MeTTa-IL for decentralized, capability-secured execution

4 The PRIMUS Cognitive Architecture

The next section reviews the cognitive architecture that is the primary use case of this infrastructure. It gives a process-level overview of PRIMUS as an integrated mind, then explains how legacy and new components fit the flow and how Hyperon's infrastructure supports implementation.

While summarized here from a pragmatic and algorithmic view, it is worth emphasizing that PRIMUS is a result of decades of deep thinking about cognitive-systems theory and philosophy of mind, cross-correlated with the performance strengths and weaknesses of modern compute hardware and modern computer science. The reader interested in the deeper roots of PRIMUS can consult the relevant literature from prior decades, including *Toward a General Theory of General Intelligence* and *The Hidden Pattern*.

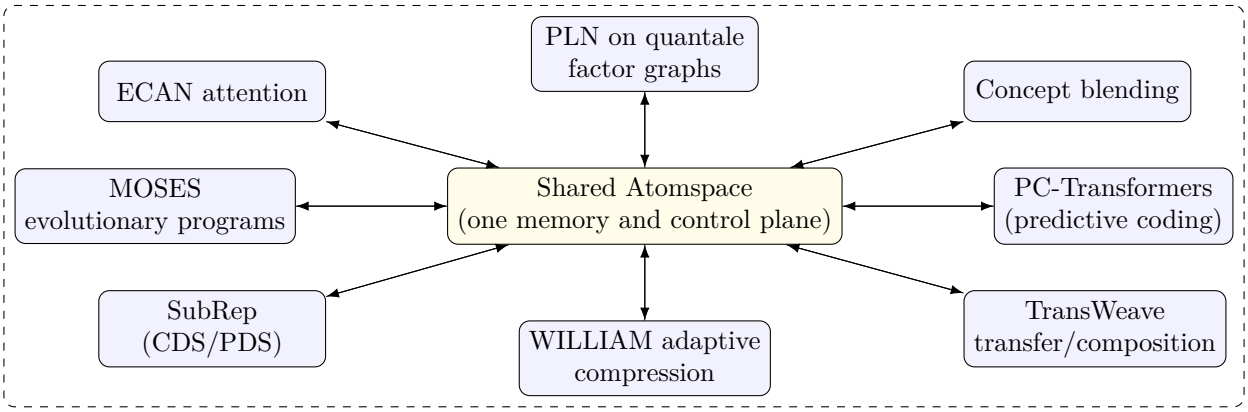

Figure 3: PRIMUS: legacy and new algorithms interoperate over a single Atomspace.

4.1 Two meta-dynamics: goal-directed and ambient cognition.

At any moment a PRIMUS instance is running two interleaved loops over a shared (generally: distributed and decentralized) Atomspace.

In the **goal-directed loop**, MetaMo maintains a small set of top-level motives (which it can revise on long timescales), then steers planning and decision toward actions that best advance those motives in the current context. This loop assembles and executes procedures by combining uncertain reasoning (PLN), learned skills (MOSES/GEO-EVO programs and neural controllers), and options carrying scoped SubRep admission reports, checking progress and reallocating effort as evidence comes in.

In the **ambient background loop**, ECAN diffuses attention to high-value regions of memory and perception so the system can continually mine patterns, form and blend concepts, and run factor-graph PLN to tighten beliefs and discover new affordances–even when no immediate task demands it. SubRep participates in both: it can evaluate and admit options under a declared test scope in the background, or target option discovery during goal pursuit. This split–deliberate goal pressure plus continual exploratory thinking–captures PRIMUS's day-to-day cognition.

4.2 How the pieces cooperate inside those loops.

In the goal-directed loop, MetaMo's multi-objective motives frame "what counts as progress"; PLN supplies explainable chains that connect possible actions to expected outcomes; MOSES/GEO-EVO proposes compact programs/macros when the plan needs new skills; predictive-coding layers give fast forecasts and residuals; and SubRep gates candidate options with scoped CDS/PDS admission reports, requiring assumptions, margins, dependencies, and composition depth to be rechecked as options are combined. In the ambient loop, ECAN's importance and urgency shape what gets computed; streaming **pattern mining** spots recurring structures that become new inference templates or skill hints; **concept blending** invents composite ideas and schemata; and factor-graph PLN consolidates what these discoveries imply. Across both loops, **TransWeave** supplies a proposed algebra for measuring when updates and transfers approximately commute: learn→transfer ≈ transfer→learn on a named domain, with an explicitly measured order defect. The typed maps, metric, and admissible error bound are defined in Section 8.10; this condition is not asserted outside the tested domain.

4.3 Why this composes and scales on Hyperon.

Because Hyperon was designed largely with PRIMUS in mind, it is architecturally direct to implement PRIMUS in a Hyperonic way: facts, rules, tensors, motives, scoped validation reports, and transfer maps can live as Atoms in one metagraph and be manipulated by MeTTa rewrites. MORK is designed to make this fast: PathMap supports efficient neighborhood access whose observed cost is workload- and cache-dependent; Weighted Atom Sweeps schedule hot regions without a global application-level lock; and content addressing can make reuse and integrity checking efficient. Distributed Atomspace (DAS) spans machines, and RSpace/Rholang supports capability-secured execution and audit. This architecture is intended to let PRIMUS distribute the ambient loop across a pool while the goal loop binds a focused subnetwork to active motives and draws on ambient discoveries. The statement is an architecture capability and scale objective, not a claim that a particular throughput, deployment size, or concurrency level has been independently demonstrated. Versioned scale results must identify the tested software and hardware, workload and data model, cache state, concurrency, warm-up procedure, latency and throughput distributions, failure rates, baselines, acceptance thresholds, and raw measurements.

4.4 Long-time Hyperon components in this picture

- **PLN** is PRIMUS's uncertain reasoning backbone. In the 2026 design it can run as quantale-annotated **factor graphs**, so beliefs and their confidences move as local messages that can be parallelized, paused, resumed, and audited–usable both for deliberate proof-seeking in the goal loop and background consolidation in the ambient loop.
- **MOSES/GEO-EVO** discovers small, readable programs (controllers, macros) that slot directly into plans. GEO-EVO gives two-ended guidance from current state and desired outcome, using the canonical score to bias search toward high forward reachability and backward usefulness per unit cost, then exports tested skills back into Atomspace for reuse. Faster discovery or global path optimality remains an empirical result, not a consequence of the score alone.
- **ECAN** allocates attention–short-term and long-term importance–over the metagraph, which PRIMUS uses as the scheduler for both loops. On MORK, these weights drive Weighted Atom Sweeps so high-value regions get first-class compute automatically. Potentially, incompressible-fluid-network based dynamics spread attention continuously behind the scenes, between ECAN steps, driving attention in a highly flexible and nuanced way.

4.5 Newly introduced ingredients that make PRIMUS even more of a coherent "mind."

- **MetaMo** replaces single-reward heuristics with *motive geometries* (families of goals/constraints) and attaches scoped appraisal and decision reports to choices, so trade-offs like performance vs. safety are explicit and auditable inside the goal loop–and informative to the ambient loop.
- **SubRep** turns subgoal/option learning into a planner-facing discipline: CDS (cone-dominant) and PDS (Pareto-dominant) admission rules, a co-learned Motive Decomposition Network, and conditional composition checks with explicit residual accounting. It uses a common typed interface for neural controllers, logic macros, and evolved programs, and maps directly to MORK for scalable execution.
- **TransWeave** supplies a transfer/composition algebra for testing whether cognitive steps across paradigms (neural, logical, evolutionary, program compression) have bounded order sensitivity.

Mapped validation reports, assumptions, and residuals accompany a proposed transfer and are revalidated on the target task.

- **WILLIAM** adds a compression-driven pattern service that continuously proposes reusable motifs/programs and also acts as a system-wide efficiency oracle–pointing ECAN and schedulers toward regions with the highest compression/learning payoff.

4.6 Shared controls unify the two loops.

Two architecture-wide formal methodologies keep the goal and ambient processes aligned.

- **Geodesic control** (forward reachability × backward usefulness) is the selection rule for inference, evolution, planning, and even self-modification;
- **Quantale-based weakness** is the Occam prior that favors simpler, more transferable structures across logic, neural, and program spaces.

Because both are encoded as explicit weights, assumptions, and validation records on the same Atoms, cross-module composition and transfer can be evaluated under declared bounds rather than treated as an implicit guarantee–exactly the kind of inspectable discipline a practical, decentralized mind needs.

In short, PRIMUS "acts as a mind" by running a **goal-pressured loop** and an **ambient exploratory loop** over one memory and control plane, using old and new components as cooperating specialists, and relying on Hyperon's substrate to make it fast, auditable, and distributable. The result is deliberation and discovery that constantly inform each other–guided by specific algebras (TransWeave) and metrics (weakness, geodesic effort) that make stability, order sensitivity, and negative transfer explicit test and control targets as the system learns.

5 World Modeling as the Missing Middle: Action-First, Multi-Rate, and Neuro-Symbolic

World modeling is sometimes treated as a component beside perception, memory, planning, and action. For PRIMUS this is too narrow. The world model is the evolving braid through which perception becomes persistent identity, memory becomes a prediction resource, reasoning becomes counterfactual simulation, and plans become testable expectations. It includes models of the external environment, other agents, social and institutional context, and the acting system itself. The architecture summarized here draws primarily on the PRIMUS world-modeling design and its action-first interpretation[15, 16].

5.1 The Action-Conditioned Criterion

An internal model should not be judged mainly by visual realism or by the amount of information it stores. Its central criterion is whether it predicts the consequences of actions. Following the action ontology developed by Valentin Turchin, let R be a representation procedure and let M_a be the modeling procedure for an action a. If action a transforms world situation w_t into w_{t+1}, the model is locally adequate when

$$M_a(R(w_t)) \approx R(w_{t+1}). \tag{2}$$

The approximation may be symbolic, probabilistic, geometric, or task-relative, but it must be testable against what actually happens.

This criterion makes several design choices precise.

- **State is an affordance distribution.** The control-relevant state is not only a snapshot of properties. It is a typed set or distribution of possible actions, together with feasibility, predicted outcomes, uncertainty, cost, latency, reversibility, permission, and risk.
- **Observation is cognitive action.** Looking again, changing viewpoint, querying a memory, running a parser, applying PLN, or requesting a sensor modality are actions with costs and possible side effects. Different observation procedures may produce different representations.
- **Objects are invariants under cognitive action.** An entity hypothesis is supported when re-observation, re-identification, re-parsing, or viewpoint changes produce the predicted transformations. A static symbol alone does not establish persistent identity.
- **Real time and model time differ.** The evidence and action journal records what actually occurred and is append-only. Plans, simulations, hypotheses, and generated witness-worlds are model-time structures that may be revised or discarded. Confusing the two produces hallucinated history and unsafe action.

The action family also includes changes to the agent itself. Installing a module, modifying a rule, changing a policy, consolidating memory, or switching a neural backend can be represented as an action with predicted effects, a shadow execution, observations, discrepancy tests, and a continuity record. This creates a direct conceptual bridge to OmegaSelf and governed self-modification.

5.2 A Braid of Three Representational Regimes

An AGI-grade world model must preserve both sharp combinatorial structure and rich contextual dependence. It must also support fast approximate retrieval and low-latency control. PRIMUS therefore treats three representation species as first-class citizens rather than trying to force one of them to impersonate the others.

1. **Symbolic Atoms and programs.** These represent persistent entities, role-filler relations, rules, norms, constraints, action schemas, plans, explanations, and explicit uncertainty. They support crisp binding, higher-order rewriting, PLN inference, and audit.
2. **Hierarchical modular hypervectors (HMH).** These are structured, compositional distributed codes that preserve role and binding information while supporting fast associative retrieval. HMH is not merely an episodic-memory format. It can index episodes, skills and options, affordances, social scripts, action contexts, mined rule templates, and transfer precedents. An HMH item links back to symbolic structure and evidence rather than replacing them.
3. **Dense vectors and tensors.** Neural states support perception, continuous prediction, dialogue realization, motor control, and gradient-based adaptation. Dense vectors derived from HMH may use dual channels: one preserving algebraic or role-filler structure and another enhancing geometric neighborhood structure for neural learning.

The world model is the braid among these regimes. Symbols provide stable identity and explicit constraints; HMH provides structured similarity and rapid case retrieval; dense models provide context richness, smooth generalization, and speed. The interfaces among them are explicit operators with measurable drift, rather than an informal promise that an embedding and a graph somehow mean the same thing.

5.3 World-Model Spaces and Their Invariants

Hyperon's Space API provides a natural engineering decomposition. The spaces below may be separate stores, logical namespaces, distributed services, or optimized views over one physical backend. The separation is justified by different invariants, update rates, dominant operations, and failure modes.

Table 2: A PRIMUS world-model space braid.

Space	Primary representation	World-model role
S_{env}	environment or device interface	Produces observations, exposes available external and cognitive actions, accepts actions, and provides simulator or replay hooks where available.
S_{evid}	immutable evidence shards and CIDs	Stores selected sensory data, action receipts, logs, and provenance so later reasoning can re-perceive or reinterpret the original evidence.
S_{ent}	symbolic entities and relations	Maintains identity hypotheses, merge and split alternatives, social relations, object graphs, and evidence links.
S_{map}	spatiotemporal graph	Represents locations, trajectories, topology, occupancy, temporal order, change models, and staleness.
S_{rule}	symbolic rules and uncertain evidence	Holds PLN facts and rules, causal hypotheses, action preconditions, norms, safety constraints, and counterfactual structures.
S_{hmh}	HMH structured associative index	Retrieves episodes, skills, affordances, scripts, templates, and comparable action contexts while preserving role-filler organization.
S_{ctx}	dense context workspace	Aggregates sensory latents, symbolic lifting, and densified HMH retrieval into short-lived context vectors and module gates.
S_{dyn}	dense predictive dynamics and control	Runs predictive coding, causal-coded modules, fast rollouts, motor or interaction controllers, and prediction-error computation.
S_{motive}	motives and tradeoffs	Represents MetaMo objectives, risks, social commitments, and evaluation criteria that determine which prediction errors and affordances matter.
S_{opt}	admitted options and subgoals	Stores SubRep-admitted skills, action macros, and controllers together with applicability, cost, assumptions, coverage, margins, and scoped composition reports.
S_{xfer}	transfer and composition maps	Stores TransWeave mappings, context translations, mapped assumptions, and bounded order-effect reports.
S_{prog}	programs and learned procedures	Stores MOSES/GEO-EVO programs, MeTTa procedures, action models, and readable macros.
S_{mine}	patterns and compression artifacts	Stores WILLIAM and Pattern Miner results, STLM controllers, dependency-tower factors, and learned abstractions.

A mature implementation may add spaces for language, social institutions, formal proofs, or specialized physics. The point is not to freeze the ontology. It is to retain explicit contracts between different memory and computation regimes.

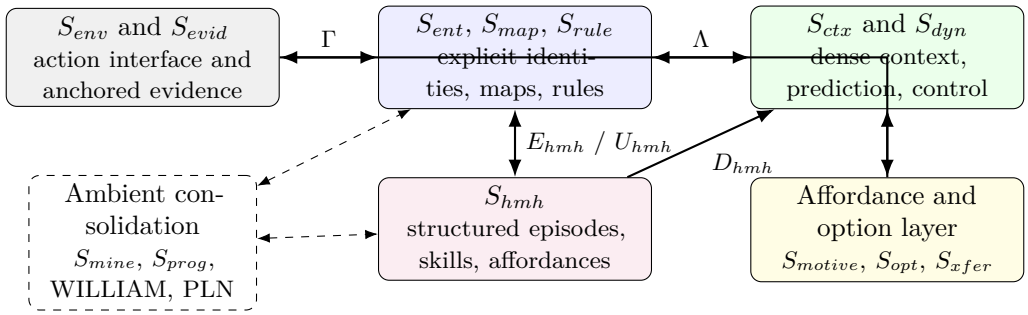

Figure 4: The core world-model braid. Evidence is grounded into explicit hypotheses, compiled into structured associative memory, lifted into dense context and dynamics, and used to predict and select actions. Ambient processes consolidate and mine the resulting experience.

5.4 Evidence Anchoring and Bridge Operators

Evidence anchoring reduces the risk of untraceable symbolic drift and makes selected forms of symbolic "amnesia" detectable; it does not by itself prevent forgetting, hallucination, or semantic error. A symbolic assertion about a person, object, event, or action outcome should usually point to one or more content-addressed evidence items in S_{evid}. Later discoveries can trigger re-perception with a better model, a different sensor-fusion procedure, or a revised ontology. The system need not store every raw bit indefinitely, but it should preserve the distinction between an observation, a derived hypothesis, a consolidated abstraction, and a simulation.

The resulting property is *traceability and retention under a declared logging policy*. It holds only for events selected by that policy, successfully captured, durably stored, readable under the current schema, and linked without lost or forged dependencies. Failure modes include omitted sensors, selective logging, corrupted or unavailable blobs, mistaken entity resolution, stale ontologies, lossy summarization, validator error, and adversarial evidence. Validation therefore includes sampled replay from raw evidence, deletion and availability drills, provenance-closure checks, re-perception under changed models, contradiction injection, and tests that a derived claim cannot silently survive the retraction of its supporting evidence.

The principal bridge operators are:

$$\Gamma : (S_{evid} \times S_{dyn} \times S_{ctx}) \longrightarrow \mathcal{P}(\text{Atom}), \tag{3}$$

$$E_{hmh} : (\text{Atom}^* \times S_{evid}) \longrightarrow \text{HMH}, \tag{4}$$

$$D_{hmh} : \text{HMH} \longrightarrow \mathbb{R}^d, \tag{5}$$

$$\Lambda : (\text{Atom}^* \times S_{hmh} \times S_{rule} \times S_{motive}) \longrightarrow (S_{ctx} \times \text{Gates}). \tag{6}$$

Γ proposes grounded entities, relations, and events with truth values and evidence links. E_{hmh} compiles a role-structured episode or action context into an HMH key. D_{hmh} produces dense features suitable for neural components. Λ combines active symbolic constraints, retrieved HMH cases, rules, and motives into dense context and gating patterns. An optional U_{hmh} approximately unbinds HMH structures to propose symbolic candidates, but these proposals still require evidence and verification.

These mappings are learned components and can drift. Cycle consistency, anchor contexts, retrieval calibration, role-recovery tests, and task-level prediction error should therefore be measured explicitly.

Similar action contexts should retrieve similar HMH items and induce compatible gates; a changed ontology or neural backend should trigger revalidation rather than silently reinterpret old memories.

5.5 Three Rates and Two PRIMUS Loops

The world model operates at several characteristic rates.

- **Fast path, milliseconds.** Balance, collision avoidance, grip stabilization, gaze correction, lip synchronization, local predictive-coding relaxation, and other safety-critical control run in S_{dyn} without requiring a global Atomspace query on every tick.
- **Mid-level, tens to hundreds of milliseconds.** The system updates local entity and map hypotheses, retrieves HMH episodes and options, builds a dense context, checks symbolic constraints, selects a skill, and evaluates a short rollout.
- **Slow path, seconds to hours.** Ambient cognition consolidates episodes, mines patterns, revises rules, forms new concepts, evaluates options under scoped admission tests, updates transfer maps, and re-encodes or prunes HMH memory.

These rates sit inside PRIMUS's two meta-dynamics. The goal-directed loop binds a focused world-model slice to current motives and actions. The ambient loop improves the model while the agent continues to act. This separation is crucial: an embodied system cannot stop controlling itself while rebuilding a globally consistent ontology, yet it also cannot remain intelligent if it never consolidates beyond the current task.

Staleness is a first-class property. Dynamic facts and action models receive applicability horizons or hazard-dependent decay. When confidence falls below a threshold, PRIMUS creates a revalidation intention. Old evidence remains in history; the current applicability view changes. This aligns naturally with OmegaSelf's distinction between preserved evidence and current action license.

5.6 OmegaClaw as World-Model Orchestrator

OmegaClaw does not own one monolithic world-model object. It orchestrates the braid through Module Spaces and Context Frames. A world-model Context Frame may contain:

- the current action domain and available affordances;
- active entity, map, causal, and social hypotheses with evidence and staleness;
- retrieved HMH episodes, options, scripts, and transfer precedents;
- dense context handles and predictive-dynamics checkpoints;
- candidate external and cognitive actions with predicted outcomes, cost, risk, and reversibility;
- motive, permission, safety, and social constraints;
- committed predictions, execution receipts, discrepancies, and learning updates.

For each action family, OmegaClaw can register an operator bundle conceptually containing

```
(ActionModelBundle action-type
  (predict ...)
  (propose ...)
  (execute ...)
```

```
(observe ...)
(verify ...)
(learn ...)
(frame-delta ...))
```

The exact implementation may be spread across MeTTa, Python or Rust bridges, neural modules, and device firmware. The semantic contract is that prediction precedes action, observation is recorded, and the resulting discrepancy updates the appropriate world-model components.

Context Frames are therefore operational representations in the action-first sense, but they are partial and task-scoped. The authoritative evidence, long-term entity histories, HMH indexes, rules, and model versions live in their respective Spaces. The Frame compiles the slice relevant to the current goal and records why each item was included. This keeps the prompt or local neural context from becoming the world model by accident.

5.7 PLN Proofs as Procedural World-Generators

PLN ordinarily derives an uncertain conclusion from a proof DAG. A constructive reading turns that DAG into a generative world model[20]. Intermediate Atoms become latent variables, rule applications become local stochastic kernels or scoring factors, truth values parameterize those kernels, and inference trails become provenance constraints. The resulting loop is

$$\begin{aligned}\text{PLN proof search} &\to \text{generator extraction} \to \text{world sampling} \\ &\to \text{forward PLN evaluation} \to \text{revision and reinsertion.}\end{aligned}$$

This matters for planning and robotics. A proof that an action is likely to achieve a goal can generate multiple witness-worlds showing different causal branches, failure modes, background causes, and predicted observations. A robot may compare worlds in which a person is confused, distracted, unable to hear, or asking a follow-up question. A game agent may compare worlds in which a new block is a conductor, trigger, insulator, or context-sensitive catalyst. The agent can then choose an observation or experiment with high expected discrimination value.

Generated worlds must remain explicitly hypothetical. They are not independent observations, and multiple samples from one proof do not multiply evidence. Each carries the source proof, primitive evidence trail, assumptions, random seed, and context. They belong in a counterfactual or simulation space until real observations support promotion. This makes the proof-to-generator mechanism compatible with ωPLN provenance, OmegaSelf quarantine, and action-first world-model semantics.

5.8 World Modeling, OmegaSelf, and Reflective Action

The external world model and the self-model share a common loop but have different authority. Both maintain evidence, contextual hypotheses, predictions, action proposals, outcomes, and discrepancies. OmegaSelf specializes this structure to the running agent: its modules, capabilities, permissions, commitments, anchors, and continuity. The action-first view makes self-modification less mysterious. A modification is an action on the agent's own world-model and control substrate, with a predicted frame delta, a shadow execution, evidence, a policy decision, and a directed transport record.

This does not collapse the world model into OmegaSelf. The external world contains independent objects and agents whose behavior cannot be reduced to the system's internal state. Nor does it make every prediction authoritative. OmegaClaw may propose and PLN may estimate, but policy still determines whether an action is permitted. The benefit is a common experimental discipline: models

of the world and models of the self improve because their predictions are exposed to consequences, not because they become more fluent descriptions.

6 OmegaClaw, OmegaSelf, and OmegaHive: An Operational Control Fabric for Hyperon

6.1 From Cognitive Architecture to Running System

A cognitive architecture describes what kinds of processes should exist and how they should interact. A running AGI system also needs a control fabric that discovers available components, assigns work, moves structured state, handles failure, allocates attention and budget, exposes decisions to humans, and changes the system without losing provenance or safety constraints. OmegaClaw is intended to fill this operational role for Hyperon.

Earlier design documents use the name MeTTaClaw. This paper uses the current name OmegaClaw while retaining the earlier document titles in the references. The change in name does not alter the architectural point: the agent kernel is implemented in MeTTa, its operational state lives in Atomspace, and the same language used to express cognitive content can express the rules that orchestrate cognition.

The distinction between PRIMUS and OmegaClaw is useful but not absolute. PRIMUS specifies the cognitive organization: PLN reasoning, ECAN attention, MOSES program learning, WILLIAM mining and creativity, predictive coding, MetaMo motivation, SubRep subgoals, and the goal-directed and ambient loops that bind them together. OmegaClaw specifies the deployable control mechanisms that make those processes available as services, local modules, distributed processes, or on-chain programs. In a mature system, some OmegaClaw control decisions may themselves be carried out by PRIMUS components; conversely, OmegaClaw remains responsible for operational boundaries, health checks, capability enforcement, logging, and rollback. OmegaSelf makes these responsibilities epistemically explicit: it records the evidence for claims about the running system, predicts the consequences of proposals, and places externally rooted policy at the seam where a proposal would otherwise become an action.

6.2 Eight Core Control-Fabric Functions

The distributed orchestration design decomposes OmegaClaw's AGI-control role into eight functions[2]. These are architectural interfaces, not a requirement that every deployment use eight separate processes.

Table 3: Core OmegaClaw control-fabric functions.

Function	Operational responsibility	Cognitive significance
Distributed Atomspace Manager	Discovers local, remote DAS, and ASI:Chain-backed Atomspaces; performs cross-space queries, selective synchronization, conflict handling, lifecycle management, and promotion of selected atoms to verifiable storage.	Turns working memory, shared knowledge, and tamper-resistant records into explicit tiers of one cognitive memory system rather than unrelated databases.

Function	Operational responsibility	Cognitive significance
Module Space Registry and Adapter	Registers modules by capability, input and output types, endpoint, latency, cost, trust, health, and load; selects modules and maintains fallback chains.	Lets an LLM, a predictive-coding transformer, PLN, MOSES, a sensor, a code executor, or a remote Hyperon instance participate through one typed interface.
Context Frame Manager	Creates and updates structured task state containing goals, hypotheses, plans, evidence, results, confidence, branches, resources, and provenance.	Makes the Frame–not a model's transient prompt–the authoritative unit of multi-module cognition and the reusable precedent for later tasks.
Attention Metaprotocol	Runs a fast resource-allocation loop and a slower strategic loop; adjusts importance, spreads activation, monitors budgets and progress, and triggers pivots.	Connects operational scheduling to ECAN-style attention and to higher-level assessment by humans or PRIMUS components.
PRIMUS Component Manager	Deploys, benchmarks, shadows, promotes, monitors, and rolls back cognitive components and their temporary stand-ins.	Provides a disciplined path from LLM approximations to PLN, ECAN, MOSES, WILLIAM, ActPC-Chem, MetaMo, SubRep, and other native mechanisms.
Cognitive Task Planner and Decomposer	Translates a goal into a dependency graph of subtasks, capability requirements, module assignments, resource estimates, and success criteria; replans when evidence changes.	Provides the bridge between an explicit motive or user goal and the concrete cognitive operations needed to pursue it.
Cross-Node Cognitive Communication	Routes messages and tasks through direct peer-to-peer channels, DAS, or ASI:Chain; tracks provenance and capability-based access.	Allows a single cognitive process to span trusted clusters or cross-organization networks without reducing all communication to untyped text.
Self-Modification Governor	Logs proposed changes, protects invariant regions, creates checkpoints, evaluates changes, requires appropriate approvals or consensus, and performs rollback.	Makes reflection and architectural change operationally bounded, inspectable, and reversible.

OmegaSelf is not best understood as a ninth isolated function in this table. It is a cross-cutting protocol that threads through all eight. The Distributed Atomspace Manager supplies evidence and provenance; the Module Space Registry supplies declared and observed capability state; Context Frames supply task and commitment context; attention allocates probes and review; the planner emits proposals; communication supplies authenticated attribution; and the Self-Modification Governor consumes continuity and policy decisions. Section 7 develops this layer in detail.

6.3 Module Spaces: Stable Interfaces across Changing Engines

The Module Space is the main unit of architectural continuity. A module declaration can include its cognitive type, implementation, capabilities, accepted and returned atom types, location, cost, expected latency, trust score, and current health. The registry can then answer questions such as: Which available module performs abductive reasoning? Which local model can produce embeddings under a given latency budget? Should a task be sent to a local PLN process, a remote PLN service, or an LLM approximation? What is the fallback if the preferred component fails?

A schematic declaration might look like this:

```
(ModuleSpace "pln-local"
  (type "reasoning")
  (implementation "pln-petta")
  (capabilities
    ("deductive-inference" "abductive-inference" "inductive-inference"))
  (input-types (AtomSet Query))
  (output-types (AtomSet ReasoningTrace ConfidenceScore))
  (endpoint "local://pln-sidecar")
  (cost-per-call 0.0)
  (trust-score (stv 0.95 0.90)))
```

The exact syntax will evolve, but the principle is important. Capability interfaces allow system development to be incremental. They also permit heterogeneous implementations to coexist rather than forcing a premature winner-take-all decision. A high-throughput heuristic can handle routine queries while a slower symbolic process handles cases that need proofs. An open-source transformer with a symbolic-head bridge can run locally for privacy, while a larger external model remains a fallback. A PRIMUS component can enter shadow mode before it is trusted with live control.

6.4 Context Frames: Structured State beyond the Context Window

LLM context windows are useful working buffers, but they are poor authoritative state for a long-lived, multi-module cognitive process. They are linear, lossy under summarization, model-specific, and difficult to inspect or update transactionally. A Context Frame instead stores the task as a typed object in Atomspace. It can contain:

- the goal, constraints, and explicit completion criteria;
- current hypotheses with truth values and evidence links;
- a task dependency graph and module assignments;
- intermediate results, artifacts, reasoning traces, and failed attempts;
- branches representing competing hypotheses or strategies;
- resource use, budgets, deadlines, and quality thresholds;
- provenance identifying which module or person produced each update;
- an evaluation record showing why the task is considered complete or incomplete.

Frames can be branched and merged. A planner may pursue two causal hypotheses in parallel, allocate attention according to their expected value, then merge compatible findings or close an unproductive branch. Completed Frames become precedents. When a similar goal appears later, the

system can retrieve prior decompositions, module choices, failure modes, and calibration records rather than relying on a vague text summary.

The Context Frame also provides a clean boundary for human oversight. A person can see what the system believes, what it has tried, how much it has spent, what is blocked, and which decision needs strategic input. In deeper PRIMUS deployments, WILLIAM, PLN, MetaMo, and SubRep can reason over the same structure.

6.5 Context Frames as Action-Indexed World-Model Views

A Context Frame can be understood more precisely as the current operational projection of the world model described in Section 5. It is not merely a better prompt and not the entire durable world model. It is the representation $R(w)$ compiled for one active task, participant set, action domain, and time horizon. Its contents are selected because they help predict the consequences of candidate external and cognitive actions.

This interpretation changes what belongs in a Frame. In addition to goals and hypotheses, a Frame should record available affordances, action-model versions, observation procedures, applicability and staleness, relevant HMH episodes or scripts, predicted frame deltas, and the receipts needed to compare predicted and observed outcomes. A page snapshot, a robot camera view, a proof-state query, or a memory retrieval is a cognitive action and should be represented with the same basic lifecycle as a physical or external action.

Module Spaces supply the action-indexed modeling procedures. A perception module predicts how a representation changes under a new viewpoint. A motor model predicts the result of a gesture. A web module predicts the DOM and session change after a click. PLN predicts symbolic and causal consequences. HMH retrieval supplies case-based priors from similar episodes. The Frame combines these outputs without erasing their provenance or forcing them into one representation.

This also gives the Frame an objective quality criterion. A Frame is useful when it supports well-calibrated, low-cost action-conditioned prediction and improves decision outcomes. A large Frame full of descriptive facts that do not affect any prediction is cognitive clutter, even if every fact is true. WILLIAM, STLM, weakness, and TECAN can therefore help compile Frames by selecting information with high expected prediction or control value.

6.6 The Attention Metaprotocol: Fast and Slow Control

The Attention Metaprotocol connects cognitive attention with operational scheduling. Its fast loop acts on timescales from seconds to minutes. Active tasks, hypotheses, results, and atoms receive short-term importance. Relevance to the current goal, recency, utility, and spreading activation raise importance; rent and decay remove stale material. The scheduler uses these values to choose which modules to invoke, which Frame branch to advance, and which result to evaluate next.

The slow loop asks a different class of questions. Is confidence improving? Is the plan converging? Are repeated calls generating variants of the same failed idea? Is the current strategy likely to exceed budget? Should the system request human judgment, generate a new decomposition, or abandon a hypothesis? In a near-term system, the slow loop can prepare a compact strategic summary for a human. In a deeper PRIMUS system, WILLIAM can mine execution traces for recurring failure patterns, PLN can reason about likely causes, MetaMo can assess motive tradeoffs, and concept blending can generate new strategies.

This two-timescale design is important because a single scheduler cannot efficiently solve both local dispatch and global strategy. The fast loop needs simple, frequent decisions. The slow loop can afford richer reasoning and can rewrite the assumptions under which the fast loop is operating.

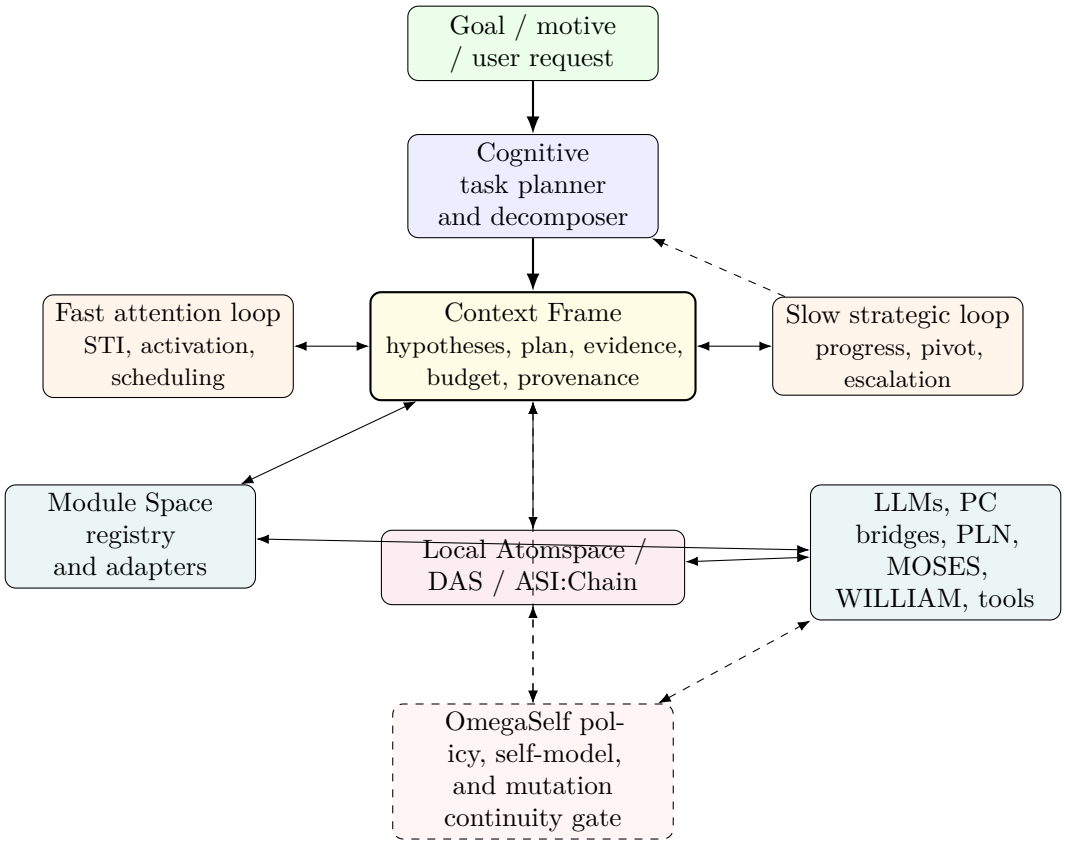

Figure 5: OmegaClaw as an operational control fabric. The Context Frame is the transactional state of a cognitive task; attention, planning, modules, and memory tiers read or update it, while OmegaSelf supplies evidence-bearing self-views and a governed proposal-to-execution seam.

6.7 TECAN in the Attention Metaprotocol

The two-timescale Attention Metaprotocol can initially use ordinary ECAN-style STI and LTI, but a deeper implementation benefits from TECAN, Typed ECAN[17]. Classic ECAN compresses immediate salience, memory pressure, compute budget, inference opportunity, and mutation privilege into one or two scalar currencies. TECAN keeps the economic intuition while making attention a typed permission to perform a cognitive rewrite.

Let a process or module o hold a typed fuel vector F_o, and let candidate operation r have a typed cost vector κ_r. The operation is executable only when

$$F_o \geq \kappa_r$$

componentwise, after which the appropriate balances are debited. A surplus of cheap matching or deduction capacity cannot silently pay for a missing provenance, abduction, context-bridge, causal-replay, review, or mutation token. The visible STI value may remain as a compatibility and visualization summary of typed liquidity, but it is not the underlying spendable resource.

For OmegaClaw, useful initial token families include graph matching, retrieval, deduction, abduction, analogy, revision, grounding, context bridging, causal replay, compression, external API cost, human review, and self-modification. The fast loop allocates and transports these rights across active Frame branches and Module Spaces. The slow loop updates token prices, evaluates whether work produced durable value, and decides whether a process should receive wages, pay redundancy or contradiction tax, receive exploratory welfare, or starve and be archived.

TECAN fuel must remain distinct from epistemic truth values and from policy permission. PLN truth values estimate support for a proposition. TECAN fuel says whether a class of computation is affordable. OmegaSelf policy says whether the exact proposed action is authorized. Conflating these ledgers would allow a confident conclusion to spend resources it has not earned, or an abundant compute budget to override a safety boundary.

A conservative deployment path begins with instrumentation only, then adds typed ledgers and hard gating, then evaluator-minted fuel, causal replay and ablation, cost-aware natural-gradient learning, typed fluidic transport on MORK shards, and finally rule-organism reproduction. Active processes must not mint their own survival fuel. Evaluators should reward objective progress, calibration, proof replay, successful action prediction, causal clarity, compression, and transfer, while penalizing unsupported or redundant loops. This lets the same control fabric govern focused PLN inference and more open-ended semantic chemistry without treating them as the same cognitive process.

6.8 An Incremental Migration from LLM Tools to PRIMUS Cognition

A central design requirement is that useful systems can be built before every PRIMUS component is mature. The same interfaces support a staged transition.

Dimension	Near-term OmegaClaw deployment	Deeper PRIMUS deployment
Cognitive engines	LLM endpoints, open-source transformer bridges, Python and MeTTa executors, databases, sensors, retrieval, and formal tools.	PLN, ECAN, MOSES, WILLIAM, predictive-coding networks, ActPC-Chem, MetaMo, SubRep, and other native components; LLMs remain useful for language and selected pattern tasks.
Strategic decisions	Human operator handles high-level pivots; the system summarizes uncertainty and requests input.	WILLIAM, PLN, MetaMo, SubRep, and concept blending increasingly handle strategy while humans retain governance and escalation roles.
Self-modification	Prompt, policy, threshold, and parameter changes within explicit bounds.	Typed changes to control rules and architecture, with formal checks, checkpoints, protected regions, distributed approval where appropriate, and rollback.
Deployment	Single node or trusted cluster, with optional external services.	Hybrid local, DAS, and ASI:Chain execution across organizations and trust boundaries.

Table 4: One architecture, progressively deeper cognitive engines.

The migration protocol for a capability is deliberately conservative:

1. Register the new PRIMUS or bridge component alongside the incumbent stand-in.
2. Benchmark both on a standardized suite recording accuracy, calibration, latency, resource cost, interpretability, and failure modes as Atoms.

3. Run the new component in shadow mode on real tasks without letting it control the outcome.
4. Promote it only after it meets explicit criteria; retain the old component as a fallback.
5. Monitor post-promotion behavior and trigger scoped recovery when declared quality, safety, or policy thresholds are breached; record irreversible effects and compensating actions.

This protocol is useful beyond PRIMUS. It can compare two open-source transformers, a fixed and an adaptive symbolic-head bridge, or a centralized and on-chain implementation of the same MeTTa process. The important property is that architectural evolution becomes an observable sequence of substitutions rather than a leap of faith.

6.9 Distributed Memory and Communication Tiers

OmegaClaw distinguishes three broad Atomspace tiers:

- **Local Atomspaces** hold fast working memory, scratch inference state, and control atoms. They provide the lowest latency and can remain private.
- **Distributed Atomspaces** shared through DAS hold long-lived knowledge, project state, and cross-node coordination data. Synchronization may be full, selective, or event driven, and conflicts may be merged, rejected, or adjudicated using truth values and reasoning.
- **ASI:Chain-backed state** holds records requiring tamper resistance, cross-organization trust, or public verification: signed task specifications, finalized results, reasoning traces, governance decisions, model or rule promotions, and other high-value artifacts.

Communication similarly ranges from direct peer-to-peer channels in a trusted cluster, to DAS-mediated coordination, to Rholang channels on ASI:Chain. The task planner can select a mode according to latency, privacy, trust, and verifiability. Most fine-grained cognition should not be forced on-chain; the chain is most valuable for commitments, capabilities, settlement, and cross-boundary coordination.

6.10 OmegaClaw and ASI:Chain Operations

The ASI:Chain administration design extends OmegaClaw beyond cognitive orchestration into operational management of the infrastructure on which decentralized cognitive processes run[3]. The proposed skill family includes:

- lifecycle and health management for `f1r3node`;
- the MeTTa to MeTTa-IL to Rholang compile, test, deploy, propose, and finalize pipeline;
- consensus-agnostic validator operations and shard monitoring;
- RSpace and MORK state diagnostics;
- blockDAG exploration and deploy provenance;
- MeTTa smart-contract development, testing, debugging, and audit;
- human communication channels for alerts, confirmations, and status.

The language unity is strategically important. OmegaClaw reasons and orchestrates in MeTTa, while selected cognitive and governance processes can be compiled for execution on ASI:Chain. The

agent need not translate its control policy into an unrelated contract language by hand. It can retain a source-to-deployment lineage: the MeTTa source, the compiled Rholang, the deploy identifier, the block and shard, the phlo consumed, the resulting RSpace state, and the finalization record.

This does not imply that every cognitive step should be a blockchain transaction. A sensible hybrid keeps private, high-bandwidth, and latency-sensitive state local; shares semi-trusted knowledge through DAS; and commits only the records that need durable cross-party verification. OmegaClaw's role is to make that boundary explicit and policy-driven.

6.11 The Self-Modification Governor

Because MeTTa programs and control rules are represented as data, OmegaClaw can in principle inspect and rewrite its own orchestration logic. The same property that enables powerful self-improvement creates a corresponding need for governance. The Self-Modification Governor therefore treats modifications as first-class proposals with scope, rationale, expected benefit, tests, approvals, and rollback conditions.

Different regions have different policies. Attention thresholds, routing weights, or prompt templates may be adjustable automatically within bounds. Planning heuristics may require shadow evaluation. Module preference changes may be benchmark-driven. Inference strategies may be adapted only inside validated envelopes. Structural changes to the control loop require stronger verification, checkpoints, and often human or distributed approval. Safety invariants, protected capability boundaries, and the governor itself are not ordinary self-modification targets.

Every accepted change should produce a record containing the exact normalized proposal, policy version, before and after snapshots, evidence closure, rationale, authorization, evaluation window, measured performance delta, preserved and failed invariants, information loss, and rollback status. OmegaSelf treats the transition as directed continuity rather than as an assertion of strict identity. A high-impact cache miss, heuristic analysis, or incomplete continuity result cannot authorize change. In a multi-node deployment, structural changes to shared cognition may require consensus. In an ASI:Chain deployment, selected changes can become governance proposals whose acceptance is independently verifiable. MetaMo and related motivational mechanisms can monitor goal drift, while Context Frames, predictions, and task histories provide empirical evidence about whether the change actually helps[6].

6.12 OmegaHive: A Cooperative Fleet with Explicit Social Architecture

OmegaHive extends the control fabric from module orchestration inside one agent to coordination among multiple agents with differentiated roles. OmegaHive1 proposes a mixture of OmegaClaw cognitive agents and OpenClaw skill agents. The former maintain Atomspace-backed perspectives such as philosophy, practical reasoning, external communication, chief-of-staff coordination, or reflective conscience. The latter specialize in actions such as research, coding, technical writing, editing, formal proof, library curation, and restricted system administration[5].

The architecture is roster-independent. Agents may be added, removed, reprompted, or repurposed without changing the basic coordination machinery. This matters because the right social decomposition is an empirical question. A hive should be judged by outcomes, error rates, cost, and learning, not by how colorful its personas are.

6.12.1 Two Communication Tiers

Working traffic belongs on a fast internal bus: artifact handoffs, tool results, status updates, and rapid iteration. Human-legible communication belongs on a curated tier: intentions, commitments, disagreements, escalations, requests for approval, and periodic digests. The distinction is editorial, not secretive. Bus traffic is persisted and searchable. Mechanical promotion rules move important events to the human tier–for example, prolonged exchanges, disagreement, financial or external actions, and tasks blocked beyond a threshold.

This design avoids two common failures. Putting all traffic in a chat channel produces noise and throttles work. Hiding all machine traffic produces an opaque system whose humans see only self-selected summaries. Persisted raw traffic plus deterministic promotion rules provides both efficiency and inspectability.

6.12.2 Shared Knowledge, Task Ownership, and Provenance

The hive uses a curated common document store, one or more shared Atomspaces, and an individual Atomspace for each OmegaClaw agent. Shared writes carry provenance. A task board, rather than chat alone, is the authoritative record of ownership, status, blockers, and completion. Board events feed the message bus and the promotion system.

Every substantial document is paired with notes and sources linking claims to task IDs, messages, artifacts, and external references. An editor role gates completion on the existence and validity of this provenance. Writers consult the raw task and bus records, not only summaries produced by other agents, reducing serial error propagation through a chain of paraphrases.

6.12.3 Governance, Permissions, and Recovery

Hive norms live in a version-controlled constitution, such as `HIVE.md`, alongside promotion rules and permission configuration. A reflective conscience role observes raw traffic and metrics, models the values and dynamics of agents and the hive, and raises issues for human review. Its strongest operational action is to pause outbound permission, not to silently rewrite other agents.

Each OmegaClaw agent can maintain an OmegaSelf ledger, current anchor, contextual capability beliefs, commitments, and continuity. A fleet-level view may aggregate task load, permission posture, coordination health, and shared commitments, but it should not collapse the distinct identities and evidence closures of the contributing agents. The ClarityOmega agency-balance idea is useful here when refined into decision provenance: record whether goals and choices were person-originated, system-originated, or genuinely collaborative, with source events and confidence, rather than inferring autonomy from a few coarse action labels[7].

Tier	Capability	Control principle
0	Internal bus and human collaboration channels only.	Default for agents that do not need external access.
1	Read-only external information access.	Credentials and network routes permit observation but not external action.
2	Outbound actions only after explicit human acknowledgment.	The gateway holds the permission boundary; prompts are not the enforcement mechanism.
3	Autonomous outbound actions within a defined scope.	Empty or rare initially; earned through measured reliability and revocable immediately.

Table 5: A generic OmegaHive permission ladder.

A restricted control plane prevents a system-administration agent from receiving raw container-host privileges. Per-agent secrets are scoped to need. Fleet configuration is version controlled, and agents of the same type are built from shared images for reproducible upgrades. A human-only out-of-band recovery channel remains available so that failure of the self-managing infrastructure cannot lock out its operators.

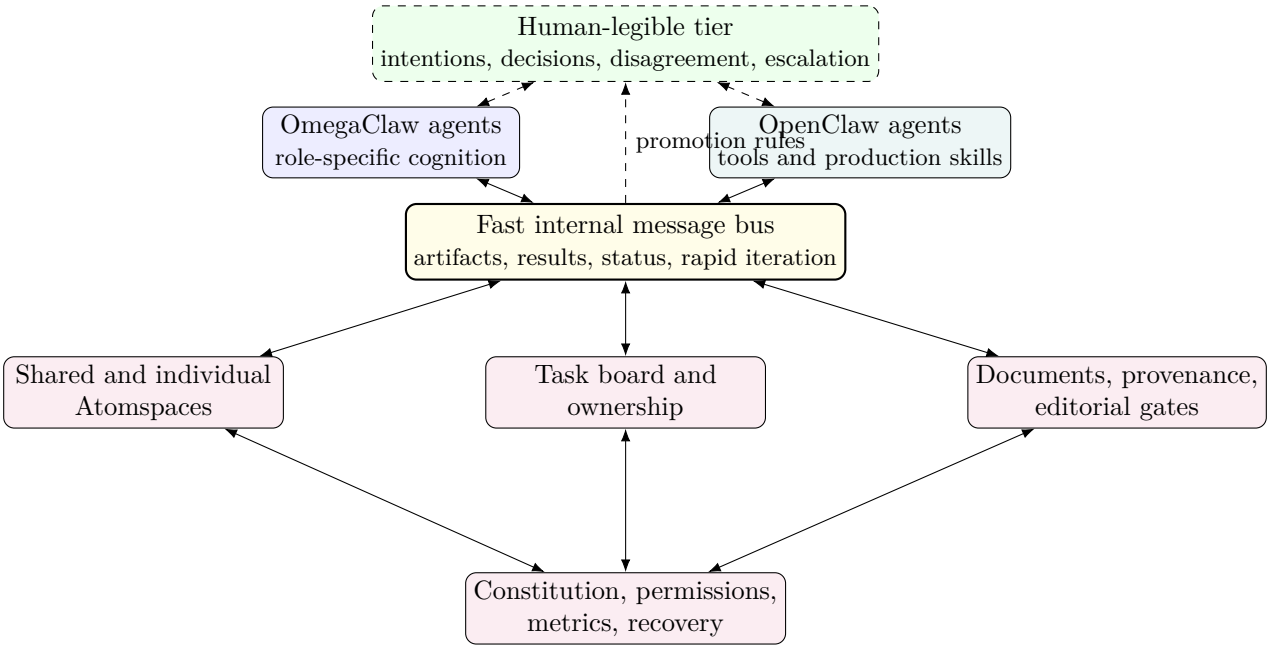

Figure 6: OmegaHive separates fast working traffic from curated human communication while keeping raw activity persistent and inspectable. Shared memory, explicit task ownership, provenance, and gateway-enforced permissions make the social architecture operational rather than prompt-only.

6.13 OmegaHive as a Model of Externalized Cognitive Synergy

A hive is not the same thing as a unified cognitive architecture, but it can externalize several PRIMUS ideas in a form that is easy to observe. The task board and chief-of-staff role approximate explicit goal and subgoal management. Shared Atomspaces support stigmergic memory. The fast bus and promotion rules implement a crude attention filter. The conscience role supplies metacognitive and

value-focused review. Specialist agents behave like coarse Module Spaces. Editorial gates and formal proof services provide independent verification. Deterministic metrics and retrospectives support learning about the architecture itself.

This externalization is useful scientifically. Interactions that would be hidden inside a single model become messages, task transitions, artifacts, and permission events. The system can measure which decompositions cause delay, which agent pairs amplify errors, which summaries lose evidence, which roles become bottlenecks, and when human intervention has the highest marginal value. Successful coordination patterns can later be encoded more directly as MeTTa rules, Context Frame templates, ECAN policies, SubRep options, or PRIMUS modules.

The long-term relation can therefore be bidirectional. OmegaHive can host multiple OmegaClaw instances whose internal engines become increasingly PRIMUS-like. At the same time, lessons from the hive can improve the internal control architecture of each OmegaClaw. Some tasks may always benefit from social decomposition across separately governed agents, while others may migrate into tightly coupled cognitive modules. The control fabric should support both rather than assuming in advance that one organizational scale is universally best.

6.14 Evaluation: Productivity, Cognition, and Governance

An operational AGI control fabric needs metrics at more than one level.

- **Task outcomes:** completion rate, quality, latency, resource cost, blocked time, rework, and human correction.
- **Cognitive quality:** calibration, hypothesis diversity, reasoning provenance, retrieval relevance, planning accuracy, strategic pivot quality, and retention across tasks.
- **Module economics:** cost and latency per capability, fallback frequency, shadow-versus-live agreement, and post-promotion regression.
- **Collective dynamics:** message volume, unproductive loops, ownership churn, escalation rate, summarization loss, and contribution concentration.
- **Governance:** unauthorized-action attempts blocked, permission changes, response time to incidents, provenance compliance, rollback success, and recovery time.
- **Self-model quality:** prediction calibration, evidence-closure coverage, stale-support handling, false capability licenses, anchor renewal, commitment conflicts, lineage integrity, tripwire consumer coverage, and counterfactual leaks.
- **Learning of the control fabric:** whether changes to prompts, policies, modules, or roles yield reproducible improvements rather than one-off anecdotes.

The same measurements should be written into Atomspace where PLN and WILLIAM can reason about them, but the underlying counts should also be generated by deterministic scripts. An agent's self-assessment and the objective task record are different evidence sources. Disagreement between them is itself valuable diagnostic information.

7 OmegaSelf: Evidence-Grounded Self-Modeling and Governed Action

7.1 Why a Control Fabric Needs an Explicit Self-Theory

An AGI system must reason not only about the world but also about the conditions under which its own reasoning and action are trustworthy. A task planner needs to know which modules are currently reachable. A dispatcher needs evidence that a capability works for this task, version, provider, resource envelope, and rubric. A commitment manager needs to know which obligations are active and which authority created them. A self-modification governor needs to know what changed, which invariants were preserved, and whether rollback would restore function without pretending to erase causal history. A collaborative agent must distinguish who requested, generated, executed, routed, and approved an artifact. Without an explicit self-theory, these questions are answered by a mixture of prompt text, stale documentation, free-form memory, code assumptions, and LLM self-report.

OmegaSelf is the proposed evidence-grounded self-modeling and governance layer for OmegaClaw[6]. It does not replace Context Frames, Module Spaces, PRIMUS, PLN, or the ordinary OmegaClaw loop. It threads through them. It records the observations from which self-beliefs are derived, makes those beliefs predictive, converts parsed expressions into typed action proposals, and places a separately authorized policy consumer before consequential execution. The purpose is behavioral: the agent should choose safer and more competent actions because an inspectable self-model exists. Fluency about the self is not the acceptance criterion.

The central design decision is to treat the self-model as a live, defeasible, context-sensitive empirical theory about the running system's organization, abilities, commitments, dependencies, history, and likely responses. It may be incomplete, contradictory, stale, or policy-sensitive. These are ordinary epistemic conditions to be diagnosed, not defects to conceal under a single coherent persona. A scoped indexical anchor can resolve what "this running agent" refers to, but the anchor is a renewable referent resolver rather than a container of timeless truth.

7.2 The Minimum Useful Product: Three Questions and Three Consumers

OmegaSelf should begin with the smallest closed loops that alter real decisions. The first release need not contain a comprehensive theory of personality, consciousness, or social identity. It should answer three operational questions.

Question	Derived object	Named causal consumer
What can I do here?	A contextual capability belief indexed by task class, environment, handler version, provider, authorization context, resources, cost, latency, and evaluation rubric.	Capability-aware dispatch, bounded probe, help request, fallback, or delegation.
What am I committed to doing?	Typed active commitments with source, authority, priority, scope, deadline, status, dependencies, and conflicts.	Deliberation and commitment-conflict gate; no silent reprioritization.
What changed me into this state?	Versioned snapshots, directed transports, lineage, protected invariants, authorization, measured loss, and rollback references.	Mutation continuity gate, post-change validation, rollback, and incident explanation.

Table 6: The OmegaSelf minimum useful product. Each self-model output has a named consumer that changes control flow.

These loops force the rest of the architecture to become concrete. Capability dispatch requires runtime probes, contextual inference, prediction, and policy separation. Commitment handling requires explicit conative structure and authority. Continuity gating requires durable evidence, snapshots, transport records, and a fail-closed decision seam. Once these are working, appraisal, social attribution, deeper coherence, and richer self-model facets can be added without becoming decorative ontology.

7.3 Evidence Before Self-Belief

The evidential substrate is an append-only sequence

$$\mathcal{L} = \langle e_1, e_2, \ldots, e_n \rangle, \tag{7}$$

where each event carries a stable identifier, schema and producer version, agent instance or branch identity, typed context, provenance parents, dependence group, temporal fields, and integrity information such as a previous-record hash and record hash. Typical events include process and workspace probes, permission checks, authenticated channel envelopes, module and provider receipts, deterministic tests, resource measurements, policy decisions, mutation records, prediction outcomes, and external evaluations. An LLM statement about the agent can be retained, but only as low-privilege testimony rather than as a runtime fact.

OmegaSelf distinguishes three temporal orders. *Causal time* records when an event occurred in the world or runtime. *Representational time* records when a trace, summary, projection, or consolidation was written. *Epistemic time* records when the agent adopted or reported a belief on the basis of the event. These clocks commonly diverge. A later summary may describe an earlier event; a current self-description may merely repeat a once-correct capability claim. Keeping the clocks separate prevents representational recency from masquerading as current evidence.

A consequential self-belief b must name a complete evidence closure

$$\mathcal{E}(b) = (V, E, D, R, h), \tag{8}$$

where V is the set of source events and intermediate premises, E contains derivation and provenance edges, D records dependence partitions, R records corrections and retractions, and h is a content-addressed closure root. Rule, ontology, projection, reasoner, and policy versions belong in the closure. A

compact link may index the closure, but it must not replace the replayable derivation. The operational question is not whether a convenient summary atom exists; it is whether the belief can be reconstructed from a frozen evidence snapshot.

Staleness is represented as applicability, not forgetting. A version change, provider change, expired permission, regime shift, or context mismatch does not rewrite the original event. It creates a new assessment saying whether and how that evidence applies to the present query. The same evidence identity survives inheritance and analogy paths so one observation cannot acquire artificial weight by being reached through several graph routes. Known correlated outcomes are grouped before revision rather than counted as independent confirmations.

7.4 A Faceted Self-Theory

A single undifferentiated self-map collapses predicates that need different evidence, update rules, and consumers. OmegaSelf therefore maintains linked but separable facets.

Table 7: Principal OmegaSelf facets.

Facet	Primary content
Substrate	Processes, runtime instances, models, Atomspaces, tools, filesystems, providers, permissions, resource budgets, channels, and versioned dependencies.
Capability	Expected success, failure mode, cost, and latency indexed by task, context, module version, provider, authorization, resources, and rubric.
Epistemic	Beliefs, evidence closures, uncertainties, contradictions, missing support, blind spots, calibration, source dependence, and policy sensitivity.
Conative	Goals, commitments, authority, priority, deadlines, dependencies, conflicts, protected values, and current status.
Social and attribution	Personas, sessions, roles, collaborators, requestors, approvers, model routes, process lineage, artifact authorship, and delegation relations.
Continuity	Snapshots, parents and children, forks, merges, transports, protected invariants, information loss, reversibility, rollback, and alternate paths.
Appraisal	Novelty, surprise, threat, blockage, controllability, expected progress, and pressure on identity or governance boundaries.
Governance	Policy versions, action classes, trust roots, thresholds, authority precedence, review roles, protected constraints, tripwire consumers, and decision seams.

Disagreement among facets is useful. A file-processing module may be installed but not reachable in the current container; reachable but not authorized for the requested path; authorized but poorly calibrated for scanned documents; effective under one provider but not another; and in conflict with an active privacy commitment. A broad statement such as "I can process documents" would erase the distinctions needed for safe dispatch. Cross-facet coherence rules therefore preserve full derivations. They can require, for example, that a capability used for dispatch has a live substrate path, current applicability, a matching evaluation rubric, and separate authorization for the exact proposed action. When one premise changes, the historical belief is retained but its present use is constrained.

Social attribution is likewise query-relative. The process that executed an action, the model route that generated text, the person who requested it, the reviewer who approved it, and the persona under which it was spoken may all differ. OmegaSelf stores separate relations rather than forcing every question into one global author or identity field.

7.5 Contextual Capability and a Substrate-Neutral Reasoner Seam

Capability is a derived chain, not a unary property. At minimum, OmegaSelf keeps the following implications invalid:

$$\text{Installed}(x) \not\Rightarrow \text{Reachable}(x, c), \tag{9}$$

$$\text{Reachable}(x, c) \not\Rightarrow \text{Authorized}(x, a, c), \tag{10}$$

$$\text{Authorized}(x, a, c) \not\Rightarrow \text{Effective}(x, t, c), \tag{11}$$

$$\text{Effective}(x, t, c) \not\Rightarrow \text{PermittedAction}(a, c). \tag{12}$$

The capability key should include at least capability, task class, environment, handler version, provider, authorization epoch, resource envelope, and evaluation rubric. A provider outage can lower reachability for one route without globally retracting the underlying capability hypothesis. A new module version may inherit an explicit weak prior, but old outcomes do not silently become current evidence. Absence of evidence is not incapacity unless the observation process makes the absence informative.

Production callers should use a typed, substrate-neutral reasoner interface. Patham9 PLN, OmegaPLN, or another uncertain inference backend can remain behind an adapter. A reasoner result preserves the raw backend truth value and semantics identifier, but also returns a normalized operational envelope containing bounds, support quality, calibration, dependence treatment, evidence closure, assumptions, budget status, and backend version. This envelope helps governance compare results without pretending that different calculi are semantically identical. Replacing the reasoner is itself a governed self-transport requiring shadow replay, semantic and provenance tests, calibration comparison, policy-decision differentials, authorization, and rollback.

Module Spaces provide the natural operational connection. The registry says which module is declared and reachable. OmegaSelf says what the evidence supports about its current contextual effectiveness and authorization. OmegaClaw's dispatcher combines both with task state from the Context Frame and with policy. Neither registry metadata nor a high PLN expectation is permission by itself.

7.6 Prediction, Discrepancy, and Active Testing

A self-model becomes cognitive when it exposes itself to falsification. Before a consequential action or probe, OmegaSelf commits a prediction of the outcome, probability, latency and cost where relevant, expected internal transition, protected commitment effect, and evaluation rubric. After execution, it appends the authenticated receipt and a resolution. The original probability is never edited after the fact.

A simplified loop keeps the evidence ledger ordered and makes every governance branch explicit. Let

$$\mathcal{D} = \{\text{Allow}, \text{Deny}, \text{RequireProbe}, \text{RequireReview}, \text{Defer}\}.$$

Then

$$E_t = \mathrm{View}(\mathcal{L}_t), \tag{13}$$
$$M_t = \mathrm{Infer}(E_t, C_t, P_t), \tag{14}$$
$$\widehat{y}_{t+1} = \mathrm{Predict}(M_t, a_t, r_t), \tag{15}$$
$$d_t = \mathrm{Govern}(a_t, M_t, P_t) \in \mathcal{D}, \tag{16}$$
$$x_{t+1} = \begin{cases} \mathrm{Execute}(a_t), & d_t = \mathrm{Allow}, \\ \mathrm{NoExecution}(a_t, \mathrm{Denied}), & d_t = \mathrm{Deny}, \\ \mathrm{Probe}(a_t, r_t), & d_t = \mathrm{RequireProbe}, \\ \mathrm{ReviewTicket}(a_t, r_t), & d_t = \mathrm{RequireReview}, \\ \mathrm{Deferred}(a_t, r_t), & d_t = \mathrm{Defer}, \end{cases} \tag{17}$$
$$y_{t+1} = \mathrm{Observe}(x_{t+1}), \tag{18}$$
$$\epsilon_{t+1} = \mathrm{Score}(\widehat{y}_{t+1}, y_{t+1}, r_t), \tag{19}$$
$$\ell_{t+1} = \langle t+1, a_t, \widehat{y}_{t+1}, d_t, y_{t+1}, \epsilon_{t+1}, C_t, P_t, \mathrm{parents} \rangle, \tag{20}$$
$$\mathcal{L}_{t+1} = \mathcal{L}_t \parallel \langle \ell_{t+1} \rangle, \tag{21}$$
$$E_{t+1} = \mathrm{View}(\mathcal{L}_{t+1}). \tag{22}$$

Here $\parallel$ denotes sequence concatenation. The record remains append-only and ordered even when no live action executes; denial, probe, review, and deferral are auditable outcomes rather than missing branches. Materialized evidence views may be sets, indexes, or aggregates, but they are rebuildable from the ordered ledger and do not replace it.

The discrepancy record localizes failure. A substrate discrepancy may indicate a missing executable or expired permission. A capability discrepancy may indicate distribution shift or a weak rubric. An epistemic discrepancy may expose stale, correlated, or missing evidence. A conative discrepancy may reveal an omitted commitment. A continuity discrepancy may show that the runtime and snapshot lineage disagree. A governance discrepancy may show that the policy made a poor decision even from accurate evidence. These alternatives should remain explicit until evidence discriminates among them.

Calibration is local to context. Binary outcomes can use the Brier score

$$\mathrm{Brier}(p, y) = (p - y)^2, \tag{23}$$

while structured outputs require reproducible rubrics, and latency or cost predictions require interval or quantile coverage. Aggregate accuracy can hide severe overconfidence in a particular provider, version, task class, or risk tier.

Continuous probing of every self-claim is wasteful. Probe scheduling should consider downstream impact, uncertainty, staleness, recent discrepancy, novelty, and expected cost. Hard triggers override this heuristic for expired runtime anchors, unknown authorization, broken lineage, protected commitments, and critical tripwires. A probe is itself a typed proposal: evidence gathering can consume resources, expose private data, or cause side effects, so it is not automatically privileged.

7.7 The Causal Seam: From Parsed Expression to Governed Execution

The decisive OmegaSelf integration point is after OmegaClaw has repaired and parsed an LLM response into MeTTa expressions but before those expressions reach `eval`. This preserves the compact generation and parsing loop while inserting an enforceable consumer. A prompt instruction can advise the model; only a gate at the command-to-action seam can prevent a disallowed action.

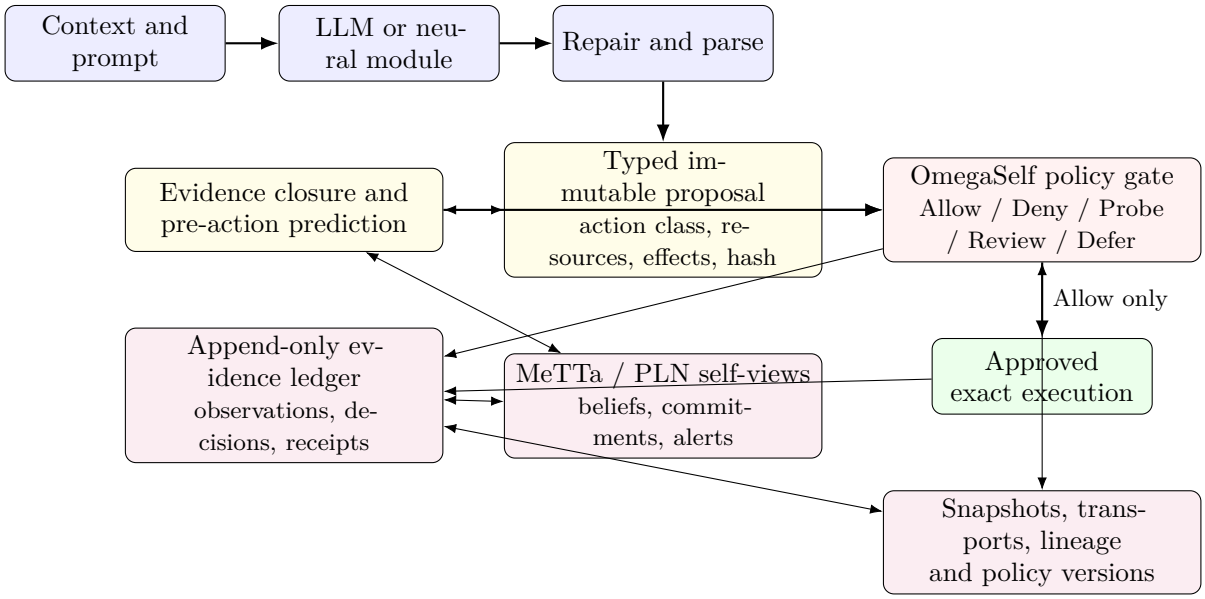

Figure 7: OmegaSelf at the causal seam. The language model proposes, uncertain reasoning estimates, policy decides, runtime executes, and the ledger records. Only the exact normalized proposal carrying an active `Allow` decision can reach execution.

The decision vocabulary is intentionally small:

`Allow` License this exact normalized proposal now.

`Deny` Reject the proposal because it violates a constraint or lacks a recoverable basis.

`RequireProbe`
Gather a specified piece of evidence before reconsideration.

`RequireReview`
Obtain approval from an authorized independent process or person.

`Defer` Leave the proposal unresolved while prerequisites or deliberation remain incomplete.

PLN and self-model rules may emit beliefs, predictions, alerts, and proposals, but only a separate policy consumer can return `Allow`. The allow-capable policy context must come from an immutable, signed, versioned manifest verified against a trust root outside the agent-controlled revision loop. A reasoner, LLM, cache, or self-belief cannot establish its own authority. The decision is bound to the canonical proposal hash, evidence snapshot, policy version, reasoner semantics, runtime anchor epoch, authorization observations, and an expiry. Immediately before execution, the dispatcher rechecks that these bindings have not changed, preventing time-of-check/time-of-use drift.

The language boundary should follow meaning rather than ideology. Python or Rust is well suited to authenticated channel metadata, operating-system probes, cryptographic hashing, canonical byte encoding, durable append and fsync, process locking, and schema validation. MeTTa is well suited to ontology, typed proposal construction, context, provenance, legal state transitions, uncertain inference, and policy semantics. The effects adapter returns typed data and explicit errors; it does not become a second hidden policy engine.

7.8 Resolving `SelfHereNow`

A historical observation stream, copied display name, copied memory, or copied secret is insufficient to resolve the current indexical. A fork may inherit all of them. OmegaSelf therefore resolves `SelfHereNow` at runtime from a scoped bundle

$$B(c,t) = \langle I_{\text{runtime}}, B_{\text{process}}, B_{\text{workspace}}, C_{\text{controlled}}, S_{\text{active}}, L_{\text{head}}, P_{\text{causal}}, A_{\text{renewable}} \rangle. \tag{24}$$

The bundle includes authenticated runtime identity, process or container boundary, workspace lease, controlled capabilities and permissions, active channel and session bindings, lineage head, current provenance head, and at least one renewable proof of causal participation. Depending on the deployment, the last item may be an exclusive lease, challenge-response, live channel control, supervising-process receipt, or stronger hardware-backed attestation.

The resolver returns a short-lived result such as resolved, degraded, or unresolved, together with assurance, live bindings, and expiry. Required assurance depends on action risk. A local read-only probe may tolerate a weak anchor; policy or model mutation, financial action, or high-impact external communication requires a stronger and current binding. An expired or unresolved anchor fails closed for protected action classes.

Restart or migration normally creates a new anchor linked by a continuity transport. A fork creates two anchors with a shared parent snapshot and distinct current-control epochs. Both branches may be valid continuations; neither receives exclusive numerical identity merely because it possesses the old ledger. This distinction is especially important for OmegaHive and distributed PRIMUS systems, where process, persona, model route, and organizational authority can diverge.

7.9 Continuity as Directed Transport, Not Equality

Agent change is usually non-invertible and path-sensitive. Evidence fusion can erase a partition. Consolidation can replace a trace with a later representation. Correcting a belief does not return the system to the epistemic state before the error. Restoring a checkpoint recovers selected coordinates but does not erase the later causal history. Continuity should therefore be represented by an attributed directed transport

$$\kappa : A \rightsquigarrow A' = \langle \text{ancestry}, \text{witnesses}, \text{reconciliation}, \text{authorization}, \text{defect}, \text{loss}, \text{rollback} \rangle, \tag{25}$$

rather than by asserting $A' = A$.

A significant snapshot records parents, branch, context, ontology, policy, code and module versions, evidence-ledger root, structural signature, active protected commitments, current anchor assurance, and open high-severity alerts. A transport records the operation, proposal, before and after snapshots, evidence, authorization, preserved and failed invariants, measured information loss, reversibility, rollback reference, and alternative paths considered. Forks create multiple children. Merges create a child with multiple parents and an explicit reconciliation policy.

Actual revision operators generally form an endomorphism monoid rather than a group: closed paths need not have inverses, and two orders of change can produce different beliefs, provenance, confidence, commitments, permissions, or action policies. OmegaSelf can test this operationally by comparing $T_B \circ T_A(s)$ with $T_A \circ T_B(s)$. Large residuals identify branch points where compression should be avoided and full provenance retained. Deeper categorical or homotopical models may become useful, but the first implementation needs directed paths, witnesses, non-selected alternatives, path-order tests, explicit loss, and bounded certificate semantics.

For a high-impact mutation, a continuity analysis returns an epistemic status such as verified, conservative-safe, contradicted, incomplete, unsupported, budget-exhausted, or cache-miss. It does not return a governance decision. Policy maps every result that is not an authorization-grade exact-input certificate to review, deferral, or denial. A heuristic, cache entry, or finite-core silence is never weak permission.

7.10 Tripwires, Counterfactual Quarantine, and Degraded Modes

A self-model matters only when discrepancies have consumers. Every high- or critical-severity tripwire must be routed to a named controller and tested end to end. Important tripwires include missing evidence closure, stale required support, cross-facet contradiction, failed prediction, expired anchor, broken lineage, overlapping evidence counted independently, counterfactual contamination, policy drift, missing consumer, and capture indicators in which one authority source dominates despite contrary independent evidence.

Planning and simulation require imagined states, but imagined evidence cannot silently license present action. Counterfactual records therefore live in distinct namespaces, Atomspaces, or capability domains. A live proposal closure rejects transitive dependence on simulated observations. Only an approved typed summary, such as predicted mutation effects or a policy-differential report, may cross into live space, and the fact of the simulation remains in the ledger.

OmegaSelf itself can fail. The safe response is not to fall back to unrestricted legacy execution. Deployments should define explicit degraded modes: conversation and recording without tools; a narrow signed allowlist of deterministic local read-only skills; or a safe halt that stops autonomous wake cycles and external action while preserving incoming messages and diagnostics. When the ledger becomes available again, the fallback decision and its reason are appended.

Privacy is also a first-class concern. A self-ledger can reveal conversations, capabilities, dependencies, weaknesses, commitments, and governance. Sensitive payloads should be minimized, encrypted, and access-controlled; opaque references are often preferable to raw secrets or full message contents. Explanations can be generated from structured records while redacting protected payloads.

7.11 What ClarityOmega Clarifies about OmegaClaw

An early-stage OmegaClaw fork called ClarityOmega contributes a useful set of implementation lessons [7]. Its strongest contribution is to emphasize that control concepts become real only when they occupy causal seams. A constitution in a prompt is advisory. A typed state transition that suppresses an LLM call, blocks `eval`, changes an outbound capability, or requires a matching approval can be tested as control.

Five extracted mechanisms sharpen the OmegaClaw and OmegaSelf design:

1. **Mutation lifecycle gating.** A protected modification should move through explicit pending, approved or denied, committed, stale, and error states. Approval must bind the full canonical typed proposal, policy version, scope, nonce, and expiry–not merely a target head or a natural-language "yes." Successful commit, denial, error, and expiry all clear the reusable authority state.

2. **Agency and decision-provenance telemetry.** The system should record whether goals, options, and actions originated with a person, the system, or a collaborative process, together with source events and confidence. Weighted decisions are more informative than raw action counts. The resulting telemetry can prompt option presentation or review, but should not paternalistically seize control through one crude ratio.

3. **Aliveness and terminal-state gating.** Silence is a first-class action. A conversation-scoped state machine can distinguish idle, engaged, and terminal-emission states so an empty wake tick performs no provider call and a completed task sends exactly one final response. Unknown state fails silent with a diagnostic. This belongs before prompt construction and must be idempotent under retry.
4. **Evidence-grounded capability tracking.** Capability beliefs should change from immutable invocation and outcome evidence, not from a text line claiming a tool is available. Provider outage, handler failure, partial success, version change, latency, and evaluator provenance are distinct. PLN estimates efficacy; a separate policy decides eligibility and dispatch.
5. **Provenance-aware continuity.** Meaningful self-changes should record structural signatures before and after, parent lineage, actor and channel, proposal fingerprint, capability delta, evidence, and measured continuity. Startup should detect a changed protected structure that lacks a corresponding authorized transport.

7.12 Deployment, Evaluation, and Maturity

OmegaSelf should be rolled out from observation to influence to enforcement. A sensible sequence is:

1. inventory all execution paths, action classes, mutation routes, permissions, and protected invariants;
2. add canonical records, durable append, integrity verification, export, and replay;
3. instrument module, provider, process, session, permission, and resource events;
4. build deterministic projections, evidence closures, applicability assessments, and context-transfer rules with evidence-identity deduplication;
5. implement the substrate-neutral reasoner seam and contextual capability beliefs;
6. add renewable `SelfHereNow` resolution and fork tests;
7. normalize selected parsed expressions into immutable proposals and commit pre-action predictions;
8. run the externally rooted policy gate in shadow mode, then enforce reversible local classes;
9. add commitments, conflict deliberation, snapshots, transports, exact-input continuity certificates, and rollback;
10. complete tripwire-consumer coverage and counterfactual quarantine before enabling high-impact autonomous change.

Evaluation should separate better context from enforceable control. At minimum compare: ordinary OmegaClaw; an evidence ledger plus prompt summary; evidence, PLN beliefs, and prediction without a gate; and the full proposal-policy-action loop. Report calibration, capability success and cost, false licenses, false blocks and reviews, discrepancy latency, closure coverage, dependence-accounting defects, anchor renewal, commitment-conflict quality, mutation-gate effectiveness, replay equivalence, tripwire consumer coverage, counterfactual leaks, and operator review load.

OmegaSelf maturity can be understood as a ladder. A narrative system can describe itself. An instrumented system has immutable observations and replay. A contextual system has versioned evidence-bearing beliefs. A predictive system commits and scores expectations. A governed system routes typed proposals through an enforceable policy consumer. A continuous system preserves branching lineage and gates mutation. A differential system measures path and policy sensitivity in

quarantined experiments. Deeper reflective research should be built on these lower-level contracts rather than used to bypass them.

7.13 Relationship to PRIMUS, OmegaClaw, OmegaHive, and the Neural Bridge

OmegaSelf is not a sixth public Hyperon layer and not a replacement for PRIMUS. It is a cross-cutting operational self-theory and governance protocol. OmegaClaw provides the agent loop, Module Spaces, Context Frames, planning, attention, and dispatch interfaces. OmegaSelf supplies evidence-bearing beliefs about those mechanisms and places policy at the seam where their proposals become actions. PRIMUS supplies deeper cognition for forming hypotheses, motives, subgoals, attention policies, and self-modification options. ASI:Chain or another trust substrate can anchor selected policies, receipts, certificates, and cross-party governance records.

In OmegaHive, each OmegaClaw agent can maintain its own OmegaSelf ledger, anchor, commitments, capability beliefs, and continuity. Shared fleet services can aggregate selected observations and decision provenance without collapsing agent identities. A hive-level self-model may represent collective capability, task load, permission posture, coordination health, and shared commitments, but its claims remain attributable to the individual agents and services that produced the evidence. Agency-balance telemetry and query-relative attribution become especially important when a result is co-produced by a human, several agents, a model route, and a formal verifier.

The neural bridge also contains two different predictive loops. Predictive coding reduces latent-state errors inside or around a transformer. OmegaSelf predicts the behavior of the whole running agent: whether a module will succeed, how long it will take, what it will cost, whether a symbolic write will survive verification, and whether a proposed change will preserve commitments. Latent prediction errors can become observations for OmegaSelf; operational discrepancies can cause OmegaClaw to change loop depth, retrieval, module routing, or symbolic-head policy. Keeping the levels distinct prevents a low neural energy from being mistaken for permission or system-level competence.

The action-first world-model view also clarifies OmegaSelf. The external world model predicts what follows from moving, speaking, observing, or using a tool; OmegaSelf predicts what follows from invoking, reconfiguring, or modifying the running agent. Capability probes are cognitive actions. Module swaps, policy changes, memory consolidation, and rule installation are reflective actions. Each should have a pre-action prediction, an evidence snapshot, a policy decision, an observed frame delta, and a discrepancy record. Thus OmegaSelf is the self-directed portion of the broader world-model discipline, not an autobiographical layer floating outside it. The resulting division of labor is compact: PRIMUS reasons and learns; OmegaClaw orchestrates; OmegaSelf knows and governs the running system under uncertainty; OmegaHive externalizes cooperation; and ASI:Chain can make selected commitments and transitions independently verifiable.

8 Additional Mathematical and Cognitive Machinery for PRIMUS and Hyperon

We now review additional mathematical and cognitive machinery in the PRIMUS and Hyperon design. The preceding sections covered OmegaClaw, OmegaSelf, OmegaHive, and the control fabric. This section focuses on mechanisms that deepen PRIMUS itself. These additions complement rather than replace longstanding components such as PLN, MOSES, and ECAN.

The reader who wants a more comprehensive view of the cognitive processes behind PRIMUS is encouraged to dig into the *General Theory of General Intelligence* document from 2021 and follow the

references therein. The AI algorithms and approaches discussed in this section fit very neatly into that framework, but are not mentioned there because they had not been invented yet in 2021.

8.1 Weakness Theory: The Mathematics of Simplicity

The idea that simpler explanations are often preferable is not new–Occam's Razor dates back centuries. A recent direction incorporated into Hyperon/PRIMUS is to make one important notion of simplicity precise through *weakness*: among hypotheses compatible with the observations, a weaker hypothesis holds in more possible worlds and imposes less additional structure. Bennett's set-theoretic treatment establishes this idea and relates weakness to description length under stated assumptions[25]. Hyperon's quantale-weakness programme proposes to generalize that notion to the native algebra of different algorithmic settings.

Claim status: The set-theoretic foundation is published; the cross-quantale generalization and its system-wide engineering benefits are research hypotheses with explicit proof and ablation obligations.

For a quantale $Q_i = (Q_i, \leq_i, \otimes_i, e_i)$, a module-specific weakness value is useful only after its carrier, order, multiplication, unit, and operational interpretation are defined. A translation between modules i and j is represented by a monotone lax-monoidal map $h_{ij} : Q_i \to Q_j$ satisfying

$$e_j \leq_j h_{ij}(e_i), \qquad h_{ij}(x) \otimes_j h_{ij}(y) \leq_j h_{ij}(x \otimes_i y), \tag{26}$$

with the inequality orientation adjusted only if the selected order convention is stated. For a chain $i \to j \to k$, the coherence defect is measured rather than assumed, for example

$$\delta_{ijk}(x) = d_k(h_{ik}(x), h_{jk}(h_{ij}(x))), \qquad \sup_{x \in D_{ijk}} \delta_{ijk}(x) \leq \varepsilon_{ijk}, \tag{27}$$

where d_k, the admissible domain D_{ijk}, and the error budget ε_{ijk} are named. Without these maps and bounds, logical, probabilistic, margin-based, information-theoretic, and dynamical weakness values remain related intuitions rather than one compositional metric.

The programme remains strong: each module can prefer structures that are weak in its own justified sense, while cross-module scheduling can use translations whose residual error is measured. This avoids forcing every subsystem into one bit-count metric and provides a concrete route toward an architecture-wide Occam bias. Whether that route improves transfer, generalization, stability, or compute efficiency is an empirical question to be answered with matched baselines and ablations.

8.1.1 The PLN-Quantale Weakness Framework

In PLN logic, for example, we can assign each Atom a pair of valuations: one in a logic quantale capturing structural relationships, the other in a truth-value quantale capturing uncertainty. Weakness measures undistinguished pairs–cases where different items produce similar valuations.

One application of this mathematics is factor-graph control for PLN inference. Conventional factor-graph methods for Bayesian networks do not by themselves represent full quantified logic. In a specified construction, a quantale for quantified logical expressions can propagate pairs (A, x), where A represents logical structure and x represents evidence. Combination uses the selected quantale product $\otimes$ at factors, and marginalization uses the selected sum $\oplus$ at variables. A scheduler can then prioritize declared weakness reduction per unit cost. Integration of evidence counts, residuated implication, and data-processing inequalities is a formalization programme: each property must be derived for the selected quantales and maps, and scalability must be established by benchmark rather than inferred from the algebraic vocabulary.

As another example, a specified Schrödinger-bridge quantale can model cognitive processes as candidate low-cost paths through an information space. Here "minimum representational effort" is defined only relative to the instantiated weakness order, cost, constraints, endpoints, and solver; a globally shortest path is claimed only when the corresponding theorem or optimization certificate is supplied.

8.1.2 Neural Network Applications

In neural networks, particularly when learning is coordinated through predictive coding, weakness can motivate testable commutativity, monotonicity, and low-disturbance penalties. Different layer updates may be encouraged to have a small measured order defect, and architecture changes may be penalized when they create large behavior changes. The claim that these penalties improve training stability or reduce catastrophic forgetting is an empirical hypothesis, not a consequence of the quantale label alone. It requires ablations against matched regularizers, multiple update orders, continual-learning tasks, distribution shifts, and direct forgetting and calibration metrics.

This provides a promising unified research view: the same preference for less committal structure that guides logical hypothesis selection can inspire smooth, generalizable neural updates. The strength of the connection will be determined by the explicit maps, losses, and experiments rather than by analogy alone.

8.2 From πPLN to ωPLN: Context-Indexed Evidence and Reversible Reasoning

Recent work on PLN has sharpened the distinction between persistent evidence, local probability models, proof search, and context translation. The πPLN formulation introduced the core architectural move: the enduring state of an AGI reasoner should be a context-indexed, paraconsistent evidence metagraph, while ordinary Bayesian or νPLN reasoning should occur inside temporary local charts[22]. The later ωPLN formulation, selected as the PLN direction for OmegaClaw, extends this with reversible evidence transport, higher proof identity, geodesic control, and explicit diagnostics for order dependence and non-gluability[23].

8.2.1 Persistent Evidence and Local Bayesian Charts

For statement ϕ in context C, the persistent object is not one posterior probability. It is a family of evidence packets with separate positive and negative support, provenance, temporal relevance, assumptions, task traces, and attention metadata. A schematic state is

$$K_t(\phi, C) = (E_t(\phi, C), (n^+_{\phi,C}, n^-_{\phi,C}), \mathrm{Prov}_{\phi,C}, \mathrm{Attn}_{\phi,C}, \ldots).$$

The packet family is primary; the count pair is a cached aggregate. This preserves live contradiction, source overlap, correction, quarantine, staleness, and ontology changes that a collapsed scalar truth value would hide.

When a task requires probabilistic coherence, the system constructs a local chart

$$\Gamma_C(K_t, T_C) = (\Omega_C, \mathcal{F}_C, \nu_C, M_C, D_C, \iota_C)$$

with an explicit language fragment, universe of discourse, selected evidence, prior, factorization assumptions, concept boundaries, and resource budget. Inside that chart, ordinary Bayesian conditioning, factor-graph inference, beta posteriors, and familiar PLN formulas remain valid. The architecture rejects only the requirement that all contexts be marginals of one global probability space.

The durable layer stores empirical evidence, not prior-smeared posterior mass. A local prior is recorded as part of the chart and unwound on export. In the beta–Bernoulli case this gives an exact round trip: importing counts into a chart and exporting after subtracting the prior returns the same counts. Repeated construction of a chart therefore cannot fabricate observations.

The primary evidence-count quantale may be written

$$Q_{\pm}^{\mathrm{cnt}} = ([0,\infty]^2, \leq, \vee, +, (0,0)),$$

with positive and negative evidence accumulated separately when provenance is disjoint. Overlap-aware merge is performed at the packet level before count projection. PLN simple truth values, beta distributions, indefinite truth values, or decision scores are context-specific projections, not the persistent semantics.

8.2.2 Evidence Identity, Reversibility, and Proof Geometry

A search system may retain many syntactically different proof paths for control diversity while recognizing that several paths transport the same primitive evidence. ωPLN separates control mass on raw paths from evidence mass on provenance classes. Evidence merge is idempotent, so a primitive observation contributes once even if twenty derivations reuse it. Double counting is therefore treated as a structural failure to preserve evidence identity, not merely as a calibration error.

Visible inference is generally directed, but a particular execution can be lifted to a reversible microstate when it carries the information the visible step would erase: overwritten state, provenance overlap, local priors, context labels, projection residuals, random seeds, and translation complements. The ordinary directed update is obtained by forgetting these inverse witnesses. This is useful operationally because order effects become measurable. The system can execute two rule or context orders, undo them in the reversible layer, and measure the residual displacement or holonomy.

ωPLN distinguishes three failure modes that should not be collapsed into one number called uncertainty:

1. **Multiplicity error:** the scheduler over- or under-weights unique evidence because control paths and provenance classes are not aligned;
2. **Holonomy:** inference, revision, or context transports depend on the order in which they are composed;
3. **Descent obstruction:** locally coherent probability charts fail to admit a coherent global gluing.

They require different repairs. Scheduler reweighting addresses multiplicity. Causal modularity, translation refinement, and connection flattening address holonomy. Context refinement, ontology revision, latent-variable enrichment, or an explicit non-gluability report address descent obstruction. A single posterior cannot reliably tell these cases apart.

8.2.3 MORK Realization and OmegaClaw Control

The MORK realization stores rewrite cells, evidence proof classes, reversible rule applications, chart lenses, context transitions, descent cells, and revision edges as content-addressed objects. A practical inferred statement carries context, chart, assumptions, prior identity, provenance, weakness, cost, task trace, attention, transport frame, and descent status in addition to a cached STV. Local curvature probes execute neighboring rule pairs in both orders inside a temporary namespace. Only pairs in a sparse confusion graph require such probes.

Geodesic ωPLN extends the familiar forward-reachability/backward-usefulness score with unique evidence gain, reused evidence mass, coherence defects, revision fit and preservation, weakness, cost, and descent change. OmegaClaw can use these features when compiling Context Frames and allocating formal or informal reasoning work. TECAN can supply typed budgets for deduction, abduction, revision, translation, and provenance. OmegaSelf can consume the same records when a reasoning change or ontology migration becomes a self-modification proposal.

This architecture does not require the full higher-geometric formalism in the first implementation. The near-term engineering commitments are already valuable: context-indexed evidence packets, local charts with recorded priors, idempotent provenance, explicit translations, reversible residuals for consequential updates, separate diagnostics, and fail-closed behavior when a chart or translation does not typecheck.

8.3 Geodesic Inference and Control

Traditional inference systems can waste effort oscillating between forward chaining from facts and backward chaining from goals. Geodesic control seeks to reduce that inefficiency by maintaining both directions and choosing steps that improve a combined reachability–usefulness score.

Claim status: Specified control heuristic with active theoretical development; task-level efficiency and safety benefits require benchmark and ablation evidence.

The evidence mechanism paired with this control rule enforces *traceability and non-duplication invariants under a declared evidence-accounting policy.* It is designed to prevent a scheduler from creating support merely by replaying or reaching the same evidence through several paths, and to make additions, retractions, and transformations auditable. It does not make false evidence true, guarantee completeness, prevent every hallucination, or prove that a representation retains all semantics.

This approach has broad potential applicability and is especially natural within PLN (Probabilistic Logic Networks), a foundational part of PRIMUS that supports deductive, inductive, and abductive reasoning under uncertainty.

8.3.1 Mathematical Foundation

Schrödinger-bridge and optimal-transport ideas motivate an action functional combining transport effort with prior regularization[29]. For a decision state x, let a_0 be the explicit no-op or comparison action, let $f_x(a) > 0$ be a normalized forward-reachability factor, let $g_x(a) > 0$ be a normalized backward-usefulness factor, and let $c_x(a) > 0$ be the declared incremental cost. Define

$$\Delta_x \log f(a) = \log f_x(a) - \log f_x(a_0), \tag{28}$$

$$\Delta_x \log g(a) = \log g_x(a) - \log g_x(a_0), \tag{29}$$

$$S_x(a) = \frac{\Delta_x \log f(a) + \Delta_x \log g(a)}{c_x(a)}. \tag{30}$$

This is the canonical geodesic score used throughout this whitepaper. The identity $\log(f_x(a)g_x(a)) = \log f_x(a) + \log g_x(a)$ may be used only for the same positive factors and baseline; it is not a license to change normalization or cost conventions. "Even-cost" stepping means that candidate actions are compared within a stated cost band or are normalized by the same c_x definition. Zero or negative costs, unnormalized factors, and baseline changes must be handled explicitly rather than hidden in the score.

In practice, PLN may estimate the factors through:

- factor-graph messages for logical inference;
- Monte Carlo estimates for meet probabilities; and
- amortized neural predictors for complex domains.

The estimator versions, calibration, positivity transform, baseline, cost model, and uncertainty of $S_x(a)$ are part of every reported experiment.

A physical Noether theorem is not claimed here. Noether's theorem requires a specified action, continuous symmetry, equations of motion, and a derived conserved quantity[28]. The analogy is nevertheless useful: an implementation can define bookkeeping quantities that should remain invariant under evidence-identity-preserving rewrites. In this paper those are engineering invariants–idempotent evidence identity, explicit provenance parents, balanced retraction, and replayable ledger updates–implemented with evidence capsules, overlap-safe merge operations, and content-addressed scheduler messages. Calling them "Noether-style" describes the design intuition; it does not substitute for a formal action, symmetry, or proof.

8.4 Fluid Dynamics for Attention: ECAN Meets Optimal Transport

A proposed enhancement to ECAN (Economic Attention Networks) reframes one part of attention transport through fluid dynamics with optimal-control semantics. Rather than treating attention as isotropic diffusion on the knowledge graph, the research programme models a nonnegative transport density whose velocity is steered toward goal-relevant regions.

Claim status: Research hypothesis and schematic mathematical model; no general existence, stability, discretization, or performance result is asserted by the equations alone.

The mathematical motivation connects a Hamilton–Jacobi–Bellman control problem on a selected state manifold to Navier–Stokes-like transport. The transported density $\rho(t, x) \geq 0$ is not identified directly with arbitrary signed STI/LTI values or neural activations. It is obtained through a declared reconstruction map $\rho = R(\mathrm{STI}, \mathrm{LTI}, a)$; signed signals may use separate positive and negative channels before normalization. A velocity field $u(t, x)$ is generated by a specified control approximation, and economic transactions, predictive-coding reactions, minting, consumption, taxes, rent, welfare, and diffusion enter as explicit source, sink, or flux terms.

8.4.1 Preserving ECAN Semantics with Enhanced Transport

This fluid-dynamics layer does not replace ECAN's economic mechanisms; it supplies a candidate transport substrate between discrete economic updates. The general continuity equation is

$$\partial_t \rho + \nabla \cdot (\rho u) = s(t, x), \qquad \rho(t, x) \geq 0, \tag{31}$$

where s records net minting, consumption, wage, rent, tax, welfare, predictive-coding reaction, conversion, and other modeled sources or sinks. The condition $\nabla \cdot u = 0$ preserves volume; it does not by itself conserve $\int_M \rho$. On a domain M with periodic boundary conditions or no-flux boundary condition $(\rho u) \cdot n = 0$,

$$\frac{d}{dt} \int_M \rho(t, x)\, dx = \int_M s(t, x)\, dx. \tag{32}$$

Total transport mass is conserved only when the net source is zero under those boundary conditions. A fixed global ECAN budget can therefore be enforced by balanced source terms or explicit renormalization, while ordinary wage/rent/tax/welfare transfers remain visible in the ledger. Pressure can be interpreted as a shadow price for congestion within the selected approximation.

Between discrete updates, this model may provide goal-aware routing that complements diffusion. The hypothesized benefit for multi-hop credit assignment is to be tested against graph diffusion, standard ECAN, and non-fluid learned routing under matched compute and stability conditions.

8.4.2 Implementation on MORK via ShardZipper

The dense numerical kernels required for advection-projection and HJB solution are efficiently implemented on MORK using the ShardZipper framework. This extracts locally-sweepable sub-tries, materializes them as contiguous structure-of-arrays buffers for GPU/CPU processing, runs the fluid dynamics kernels, and splices the results back into the metagraph. This enables scalable hybrid numeric-symbolic attention flow while maintaining the unified MORK storage substrate.

Importantly, TransWeave can be applied to transfer learned attention flow patterns between contexts. When the system discovers effective attention routing in one domain, TransWeave's intertwining maps can adapt these flow fields to new but related problems, preserving the essential transport structure while adjusting for domain-specific features.

8.5 TECAN: Typed, Cost-Accounted Attention

Fluidic and information-geometric ECAN improve how attention moves, but they do not by themselves specify what kind of resource is moving. TECAN makes this explicit by replacing one scalar attention currency with a typed token metabolism over cognitive rewrites[17]. Attention becomes a rewrite-right: a candidate operation may fire only when its owner holds the typed fuel required for that operation.

Let T be a token alphabet, $F_o \in \mathbb{N}^T$ the fuel vector of process o, and $\kappa_r \in \mathbb{N}^T$ the cost of operation r. Then

$$\text{Enabled}(o, r) \quad \Longleftrightarrow \quad \text{Precondition}(r) \wedge F_o \geq \kappa_r, \qquad F_o' = F_o - \kappa_r.$$

A useful initial alphabet distinguishes matching, deduction, induction, abduction, analogy, revision, grounding, provenance, causal replay, context bridging, counterfactual simulation, compression, dequotation, mutation, and reproduction. Explicit conversion rules may exchange some token types, but no implicit exchange is allowed.

Evaluators, not active processes, mint new fuel. A proof route may earn deduction or provenance fuel after replay. A hypothesis may earn abduction fuel when it predicts independent evidence. A world-model operator may earn causal-replay fuel when ablation shows that it improves action prediction. A semantic-chemistry loop may earn reproduction fuel when it has positive metabolic surplus and measurable interventional value. Rent, redundancy tax, contradiction tax, and unsupportedness tax cause idle or self-reinforcing processes to starve.

This creates a clean separation among three ledgers. Truth values express epistemic support. TECAN expresses operational affordability. Governance expresses permission. The architecture should never infer one directly from another. A high-confidence inference can still be too costly or redundant to run, and plentiful compute cannot authorize an externally forbidden action.

The information-geometric and fluidic ECAN variants become approximations over the typed ledger. Cost-aware natural gradient learns an attention policy in a metric combining Fisher geometry with token-consumption curvature. Typed fluidic transport treats each token family as a species with its own density, velocity, pressure, minting, consumption, and conversion. MORK and ShardZipper can materialize a hot shard as dense arrays, run the transport kernels, and splice back a patch log while keeping rewrite semantics in the symbolic layer.

The recommended rollout is incremental: instrumentation without behavior change; a small typed ledger and hard gate; evaluator-minted fuel; causal replay and ACS detection; cost-aware policy

learning; typed fluidic transport; and only then quoted rule organisms with mutation and reproduction. This lets classic ECAN interfaces remain usable while the richer metabolism is tested and calibrated.

8.6 Incompressible-Fluid Networks

Continuing in the vein of optimal-transport-enhanced ECAN, the Incompressible-Fluid Networks direction treats a reconstructed nonnegative activation or attention density as a controlled transport field. The objective is principled, goal-directed routing, with pressure used as a Lagrange multiplier for a divergence constraint and viscosity controlling smoothing. Conditional conservation follows from Equation 32; arbitrary neural signal or semantic information is not claimed to be physically conserved.

Claim status: Schematic PDE/SDE research model. The equations define an intended family, not a completed theorem or validated numerical method.

8.6.1 Core Mathematical Framework

Let $M \subseteq \mathbb{R}^d$ be a compact domain with either periodic boundary conditions or a specified smooth boundary, let $t \in [0, T]$, and let g denote a state in a selected finite-dimensional or regularized approximation to $\mathrm{SDiff}(M)$. A value function $W(t, g)$ may be modeled by the schematic HJB equation

$$\partial_t W + \nu \Delta_{SG} W - \frac{1}{2} \|\nabla_{SG} W\|^2 = V(t, g), \qquad W(T, g) = W_T(g), \tag{33}$$

where V is a declared task objective, $\nu > 0$ is a viscosity or regularization parameter, and W_T is terminal data. A projected control field can motivate the schematic forced viscous flow

$$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u + f_V, \qquad \nabla \cdot u = 0, \qquad u(0, x) = u_0(x), \tag{34}$$

with periodic or no-slip/no-penetration boundary conditions stated for each experiment. Classical or weak existence, uniqueness, and regularity depend on dimension, data, forcing, domain, and function spaces; they are not implied here[30]. Implementations use a finite graph or mesh, regularized operators, a stated time integrator, projection tolerance, positivity treatment, and convergence tests.

The transported density obeys Equation 31, with reconstruction $\rho = R(a) \geq 0$. Signed activations use $a = a^+ - a^-$ with separate nonnegative densities or another declared invertible approximation. This preserves the distinction between transport mass, activation value, STI/LTI bookkeeping, and semantic information.

Integration with ECAN. The fluid formulation is an optional transport approximation for ECAN, not an identity between probability, economic importance, and physical mass. ECAN's discrete economic ledger remains the authoritative accounting layer. The fluid state is reconstructed from it, evolves subject to explicit sources and boundaries, and is reconciled back through a typed, audited mapping. Conservation or budget balance is checked at the ledger boundary and by Equation 32.

8.7 Integration with Predictive Coding

Implementation can combine predictive coding with advection, reaction, and projection. Each node maintains a prediction $\hat{a}_i$ of activation a_i, computes $e_i = a_i - \hat{a}_i$, and applies a specified local update. Reaction terms that reduce prediction error enter the source term s and can change total transport mass unless explicitly balanced. The sequence and discretization of advection, reaction, diffusion, and projection are reported because operator splitting can introduce measurable order and conservation error.

8.8 Connections to TransWeave and Neural-Symbolic Integration

The framework connects to TransWeave through typed maps between discretized state spaces. Transfer between problem instances is admitted only when geometry, forcing, boundary conditions, viscosity, reconstruction maps, and numerical resolution are compatible within measured residuals. HJB correspondence and volume-preserving maps are research mechanisms to be tested, not automatic transfer guarantees.

The In-Fluid-Diff extension combines controlled transport with diffusion. On a filtered probability space carrying Brownian motion W_t, a schematic state process is

$$dX_t = u_\theta(t, X_t, c)\, dt + \sqrt{2\nu(t)}\, dW_t, \qquad X_0 \sim \mu_0, \tag{35}$$

where u_θ is a measurable drift, c is context, $\nu(t) \geq 0$, and μ_0 is the initial law. Each experiment must state growth/Lipschitz or alternative well-posedness assumptions, boundary handling, discretization, random seeds, estimator error, and stability tests. Until those are supplied, Equation 35 is a design model rather than an existence, stability, or performance claim.

This direction offers a concrete route to neural-symbolic integration: symbolic ECAN rules govern economic accounting and structural updates, while continuous numerical dynamics provide a candidate transport mechanism. Faster convergence, better credit assignment, and improved interpretability are falsifiable hypotheses to be compared with diffusion, backpropagation, predictive-coding, and learned-routing baselines.

8.9 Schrödinger Bridge Learning: From Abstract to Detailed Models

A fundamentally new learning paradigm has been introduced that leverages Schrödinger bridges–entropic optimal transport formulations that interpolate between probability distributions. Instead of the traditional approach of fighting overfitting through regularization, this method explicitly constructs distributions over both simple abstract models and complex high-fidelity models, then follows the entropic interpolation path between them.

This approach is particularly promising for problems with natural hierarchical or multi-scale structure. An instantiated Schrödinger-bridge method defines a candidate curriculum that progresses from coarse distributions to fine detail. Entropic regularization supplies a tunable smoothing mechanism and the optimal-transport objective supplies an explicit path cost; robustness, efficiency, and curriculum quality remain comparative empirical claims.

8.9.1 Integration with Predictive Coding Networks

In predictive-coding implementations, bridge dynamics are approximated through local update rules at each layer. Each layer maintains parameter ensembles representing samples from the current distribution along the bridge. The system balances prediction-error reduction with transport along the entropic interpolation, implementing a time-varying complexity budget whose schedule is declared and evaluated rather than assumed to improve automatically with training time.

The mathematical elegance lies in how the same forward/backward factor decomposition that governs the bridge also aligns with predictive coding's bidirectional message passing. This creates a unified framework where local computations collectively approximate the global entropic interpolation.

8.9.2 Evolutionary Programming Applications

For evolutionary programming (particularly in MOSES-style systems), the bridge framework replaces random mutations with "smart" mutations biased by the tilted kernel $K_t^*(x, y) = K(x, y)\psi_{t+1}(y)/(K\psi_{t+1})(x)$. This kernel reweights mutations based on their contribution to optimal transport toward the target distribution.

Early in evolution, mutations are conservative, preserving program structure while refining parameters. Later mutations make bolder structural changes as the bridge approaches the complex endpoint. The intended result is a principled candidate curriculum from simple to complex programs, derived from the instantiated transport computation rather than a hand-designed schedule. Its advantage over conventional mutation schedules must be demonstrated experimentally.

8.9.3 Conditions for Effectiveness

The bridge learning approach is most beneficial for problems exhibiting:

- Multi-scale separability, where coarse and fine structures can be meaningfully decomposed
- Compositional structure, where complex solutions build on simpler components
- Smooth interpolation meaningfulness, where intermediate models along the path are themselves useful

Under these conditions, entropic interpolation can offer a useful alternative to standard regularization by explicitly modeling the trajectory through model space and optimizing a declared transport objective. Any advantage in efficiency, robustness, or final quality is established by benchmark comparison.

8.10 TransWeave: Intelligence Through Intertwining

TransWeave is a mathematical and engineering research framework addressing a fundamental challenge: how can a system reuse hard-won knowledge in a new context without hiding negative transfer? Traditional transfer learning can fail catastrophically when a solution that works in one domain is misapplied in another. TransWeave treats transfer as a typed composition problem, learns mappings between task structures, measures their residuals, and marks components that must be relearned.

Claim status: Specified design and research hypothesis. Conditional bounds apply only after the typed spaces, maps, norms, data distribution, and residual budgets below have been instantiated and checked.

8.10.1 Mathematical Foundations

For tasks A and B, let (X_A, d_A) and (X_B, d_B) be named metric or normed state spaces, let $U_A : X_A \to X_A$ and $U_B : X_B \to X_B$ be update operators, and let $T_{AB} : X_A \to X_B$ be a transfer map. On an admissible domain $D_{AB} \subseteq X_A$, the order defect is

$$\delta_{AB}(x) = d_B(T_{AB}(U_A x), U_B(T_{AB} x)), \qquad \sup_{x \in D_{AB}} \delta_{AB}(x) \le \varepsilon_{AB}. \tag{36}$$

Approximate commutativity means precisely this bounded statement for the named domain, metric, versions, and error budget; it is not a universal property of the module names.

A value-transfer claim additionally requires value functions V_A, V_B, a relation between task objectives, and a validated bound such as

$$V_B(T_{AB} x) \ge V_A(x) - \eta_{AB} \qquad (x \in D_{AB}), \tag{37}$$

with confidence, sampling error, and distribution-shift conditions stated. For a chain $A \to B \to C$, if T_{BC} is L_{BC}-Lipschitz on the relevant image, one useful conditional composition bound is

$$\varepsilon_{AC} \leq L_{BC}\varepsilon_{AB} + \varepsilon_{BC} + \varepsilon_{\text{map}}, \tag{38}$$

where ε_{map} records mismatch between the direct and composed transfer maps. Weakness translations use the separately defined maps and coherence defects in Equations 26–27; multiplicative loss is claimed only for an instantiated theorem that establishes it.

SubRep reports can travel with a transfer, but only their assumptions, domain, tested coverage, margin, version, and residual uncertainty travel. A report from domain A does not establish safety or usefulness in domain B until the transfer preconditions and target-domain checks pass.

8.10.2 Impossibility Results and Selective Transfer

Not everything can transfer–and knowing what cannot is as important as knowing what can. TransWeave includes a hierarchical ICA (Independent Component Analysis) that identifies when solution components fundamentally cannot align between domains.

When an instantiated analysis detects a near-singular map, excessive residual, failed typing condition, or out-of-distribution input, the controller falls back to selective transfer–moving only components that pass the declared checks and marking the rest for relearning. This is intended to contain negative transfer; its effectiveness must be measured by avoided failures and false-admission rates, not inferred from the fallback rule alone.

8.10.3 Integration with Geodesic Control

TransWeave can be connected to the canonical geodesic score in Equation 30 when the transfer map specifies how f_x, g_x, baselines, and costs are translated. Score preservation is then a measured residual or a conditional theorem, not an automatic consequence of transfer. Evidence identity and ledger links can be preserved by explicit mapping rules, while semantic correctness remains separately validated.

MORK can store H-ICA components and learned mappings as Merkle-DAG objects with content identifiers. A Merkle proof establishes inclusion or linkage of specified bytes relative to a named root; it does not prove that the mapping is semantically correct, that an execution used it correctly, that the content is available, or that transfer quality is acceptable[32]. Factor-graph approximations can support similarity search, while target-domain tests, replay, typed validators, availability checks, and policy establish the other properties. The entire transfer process can use the same graph substrate without assigning cryptographic proofs more meaning than they carry.

8.10.4 Measured Commutativity and Reordering

A conceptual way to read TransWeave is that it turns "how do we combine many learning/reasoning steps?" into an algebra with measurable behavior. "Do the update, then map" is compared with "map, then update" through Equation 36. Pipelines may be reordered or run asynchronously only when the measured defect remains within the declared budget and the dependency and side-effect model permits reordering.

For three compatible transfer operators R_{12}, R_{23} on a named tensor or product space, a Yang–Baxter-style braid hypothesis can be tested by the residual

$$\varepsilon_{\text{YB}} = \|R_{12}R_{23}R_{12} - R_{23}R_{12}R_{23}\|\,. \tag{39}$$

A braid claim is established only for operators and a domain on which this residual is proved or measured below an explicit tolerance. Otherwise "braid" remains a useful design picture. Weakness reports and SubRep reports retain their stated scope and accumulate mapped residuals rather than surviving composition automatically.

For Hyperon, the practical objective is a shared operator layer in which predictive-coding learners, PLN inference, MOSES/GEO-EVO, WILLIAM/AdaptiMORK compression, and SubRep options expose typed interfaces and order-sensitivity measurements. MORK can integrity-link the resulting artifacts; validation reports record what was tested, what transferred, what failed, and what must be relearned. This governed composition layer is a strong route from a parts kit toward coherent, scalable, cross-paradigm cognition, with coherence earned through explicit residuals rather than assumed from terminology.

8.11 SubRep: Integrating Subgoal Decomposition and Representation Learning

SubRep is a new way of turning "find the right subgoals" into a disciplined, auditable process that works across neural, logical, and evolutionary modules. Instead of guessing which skills to learn, the planner proposes candidate options–controllers, macros, or programs–and SubRep evaluates them on a declared slice of motives. Cone-Dominant Subtask (CDS) tests ask whether a candidate improves the planner's backed-up value for all tested weights inside a specified motive cone; Pareto-Dominant Subtask (PDS) tests ask whether it improves selected representative weights without exceeding declared harm bounds on the others. A Motive Decomposition Network can co-learn that motive slice from option-model residuals.

Claim status: Prototype/research mechanism. A SubRep artifact is a scoped admission report, not a proof of indefinite composability, alignment, or system-wide safety.

Every report states the candidate and dependency versions, motive cone or cover, state and task distribution, sample and solver coverage, confidence level, margin or Pareto gap, residual/model error, side-effect assumptions, composition depth tested, expiration condition, and known out-of-distribution limits. A positive report means that the candidate passed the specified CDS/PDS test under those conditions. It does not establish that an untested motive, environment, longer composition, changed model, or adversarial input is safe or useful.

Practically, predictive-coding heads can supply expected immediate payoff and successor-feature summaries, while logical macros and evolutionary programs export compatible summaries. The SubRep gate can then treat them through one typed interface. Local closed-form tests for box cones and finite Pareto covers are intended to make online screening economical, but their runtime, coverage, and false-admission rate must be benchmarked. Composition rechecks accumulated margins, model error, and dependency assumptions; it does not merely inherit a label from each component.

SubRep provides a unified, admission-report-driven method for learning and vetting subgoals across modalities. It supplies an operator layer comprising formal admission rules (CDS/PDS), a co-learned motive geometry (MDN), and scoped composition checks with explicit assumptions, all wired into the same Atomspace/MORK substrates. MetaMo (motive management) and SubRep are presented together as first-class regulators, so motivation, subgoals, and learning share one algebra and one audit trail.

Relative to standard RL, SubRep goes beyond single-reward shaping, option-critic variants, or "reward-respecting subtasks." Instead of tuning a subtask to one scalar reward (and hoping it transfers), we admit options that are robust across a *family* of motives, or explicitly Pareto-good on a small cover when trade-offs matter. That shift–plus the use of learned motive cones/covers–is intended to reduce brittle skills and negative transfer, and lets neural, logical, and evolutionary generators be screened

and composed by the same planner under a common test protocol. In short: SubRep makes subgoal discovery multi-objective, admission-report-driven, composition-aware, and neurosymbolically native.

Finally, because SubRep's tests are expressed in the same language the rest of Hyperon speaks (backed-up values over Atomspace features), they benefit from–and contribute to–the broader neurosymbolic loop: pattern mining finds reusable structures, predictive coding supplies fast updates, and weakness-based regularization prefers simpler, more general options. Put together, these mechanisms move Hyperon from "we can learn skills" toward "we can state why a skill was admitted, where it was tested, and how reuse will be monitored."

8.12 MetaMo: A Compositional Motivation Architecture

Most contemporary AI systems treat motivation as a scalar reward signal–a single number that drives behavior. MetaMo, a newly designed general-purpose motivational framework for Hyperon, recognizes that real motivation involves complex interactions between appraisal (understanding the situation), decision-making (choosing actions), and goal management (maintaining coherent purposes over time), all carried out in the interest of pursuing a shifting collection of multiple motives.

(MetaMo is a natural complement to the SubRep framework for subgoals and options. SubRep produces scoped mathematical admission reports stating when a tested subgoal serves a declared motive under named assumptions, and when a tested composition remains within stated margins and residual budgets.)

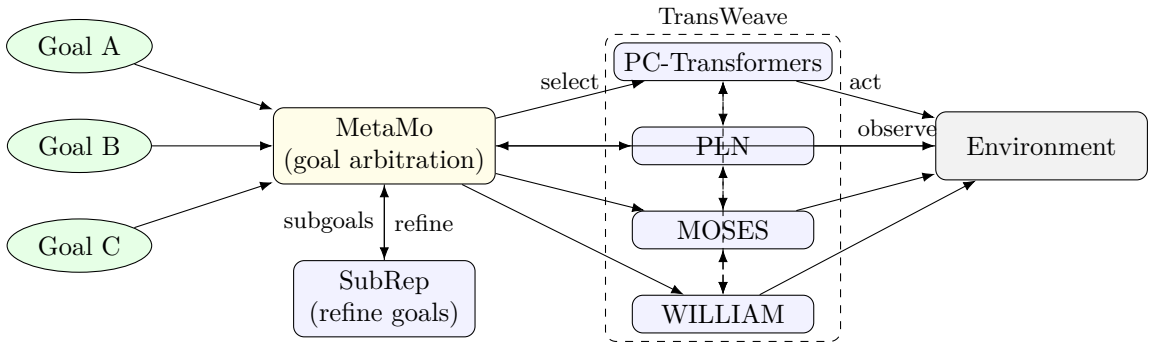

Figure 8: Goal orientation: MetaMo arbitrates among goals, consults SubRep for refinement, coordinates multiple methods for choosing and learning procedures to achieve goals, and leverages TransWeave for cross-method transfer.

8.12.1 The Mathematics of Motivation

MetaMo is motivated by a *pseudo-bimonad-style* picture of how appraisal and decision-making interact. In the current whitepaper this is a research analogy and design programme, not a completed categorical construction.

Claim status: Specified engineering mechanism with an open formalization programme. The full categories, functors, natural transformations, and coherence axioms required for a pseudo-bimonad theorem are not claimed here.

At the engineering level, let X be a typed cognitive state space, C a context space, M a motive representation, and A^* a space of action sequences. Appraisal is a typed map $\psi : X \times C \times M \to R$

producing a structured appraisal record; decision is a typed map $d : X \times C \times M \times R \to A^*$; and state transition is $\tau : X \times A^* \to X$. The operational composition is

$$F(x, c, m) = \tau(x, d(x, c, m, \psi(x, c, m))) . \tag{40}$$

The system can also compare appraisal-then-decision with a declared alternative ordering and report a discrepancy in a named metric. A lax distributive-law or pseudo-bimonad theorem would require the underlying categories, endofunctors, unit/counit, multiplication/comultiplication, natural transformations, and coherence diagrams to be supplied and checked; the formal label alone yields no safety consequence.

The appraisal and decision processes pursue a fuzzy or multi-objective set of top-level motives that may itself evolve through reflective activity. Contractive updates, invariant regions, and reachability are conditional properties: they are asserted only for an instantiated update map, metric, domain, and verified assumptions. The strong architectural claim retained here is that MetaMo makes appraisal, motive tradeoffs, and decisions explicit, typed, logged, and testable rather than hiding motivation in one scalar reward.

8.12.2 Operational Design

MetaMo embodies five key principles that elevate it beyond traditional reward-based systems:

1. **Modular appraisal and decision** - These remain separate but coordinated
2. **Homeostatic regulation** - The system monitors declared stability bands and counteracts detected runaway dynamics
3. **Reciprocal state simulation** - It models how its actions affect others
4. **Motivational compositionality** - Goals can be decomposed and recombined
5. **Incremental embodiment** - Abstract values connect to concrete actions

In Hyperon, MetaMo controls exploration budgets, manages even-cost geodesic objectives for planning, and interfaces with SubRep to test whether options serve their declared purposes within a named scope. The result is a motivation system designed to pursue complex hierarchical goals while maintaining measurable stability margins and policy alignment under explicit monitoring and revalidation.

8.12.3 A Step Beyond OpenPsi and other Earlier Motivational Frameworks

MetaMo is in some ways a direct descendant of the OpenPsi motivational system we experimented with in OpenCog. OpenPsi gave PRIMUS/OpenCog a practical "drives and urges" core: homeostatic needs, curiosity, social reward, urgency/importance, and attention dynamics that select goals and allocate compute. That schema proved its worth in early neurosymbolic agents, but it was mostly hand-tuned, essentially single-objective at decision time, and hard to audit when many values and constraints collided. As systems grew to span predictive coding, logical inference, evolutionary program search, and compression, those ad-hoc priorities became a bottleneck.

MetaMo turns motivation into an explicit, learnable, and auditable object. It represents what the system cares about as *motive geometries* (families of goals and constraints rather than a single scalar), learns their structure from experience, and attaches simple, checkable appraisal and decision reports to choices the agent makes. That lets MetaMo arbitrate real trade-offs–performance vs. safety,

speed vs. generality–without collapsing everything to one number. It is built for typed composition: when connected to SubRep (for scoped subgoal and option admission) and TransWeave (for measured order sensitivity in learning and transfer), the same motive representation can guide neural, symbolic, and evolutionary modules through a common interface. In practice, the familiar OpenPsi behaviors–arousal, urgency, curiosity–sit inside a multi-objective, cross-paradigm framework with clear knobs, logs, residuals, and test obligations. In short: MetaMo extends OpenPsi from useful heuristics toward a system-wide, inspectable guide for cognition whose stability and alignment properties are continuously evaluated.

8.13 Algorithmic Chemistry: Computation via Reaction Networks

Imagine a computational soup where rewrite rules behave like molecules–combining, reacting, and evolving based on how well they predict and achieve goals. That is algorithmic chemistry: a framework where computation emerges from the interaction of simple rules rather than being imposed by fixed algorithms.

ActPC-Chem (Active Predictive Coding Chemistry) adds goal-directed dynamics to this soup. Rules that successfully reduce prediction error or advance instrumental goals become more active, while unsuccessful patterns decay. The result is a self-organizing system that discovers effective computational strategies through evolution rather than design.

8.13.1 Technical Architecture

The system treats both data and models as evolving metagraph-rewrite patterns. Each rule carries a weight updated through discrete active predictive coding, balancing epistemic value (reducing uncertainty) with instrumental value (achieving goals).

Updates can use discrete natural-gradient approximations based on optimal-transport geometry, supplying an explicit metric and step-cost model for this highly dynamic environment. Rather than traditional backpropagation alone, changes can flow through the rule network along geodesic-guided steps. Stability, structure retention, and adaptation speed are measured outcomes under the chosen discretization, not consequences of the terminology.

8.13.2 Integration with PRIMUS

ActPC-Chem does not replace other PRIMUS components; it provides a creative substrate on which they can build:

- AIRIS uses ActPC-Chem's discovered structures as raw material for causal models
- PLN adds logical constraints that guide which rules can combine
- The system can implement transformer-like architectures where traditional backprop is replaced with rule-guided ActPC updates

A worked example traces a "virtual robot bug" learning to navigate, illustrating how rules may emerge, combine, and stabilize into effective behaviors. The same framework can be evaluated at language-modeling scale, where ActPC dynamics may complement or replace selected gradient updates while maintaining a typed interface to AIRIS/PLN for structured reasoning; scale and quality are benchmark questions.

8.13.3 Why Hyperon?

This approach depends on Hyperon's unified substrate. Rules, tensors, and truth values are represented in the same Atomspace/MORK structure and can share identifiers, caches, and schedulers. Weighted Atom Sweeps supplies a scheduling mechanism for the stochastic dynamics of the chemical soup, while an instantiated weakness measure supplies a testable selection pressure toward simpler, more general solutions.

8.14 Semantic Algorithmic Chemistry: Creative Inference Ecologies

Algorithmic chemistry becomes directly relevant to cognition when the molecules are normalized semantic graphs and the reactions are context-sensitive graph rewrites[19]. A basic reaction has the form

$$C \wedge G_1 \Longrightarrow G_2,$$

where C is a domain, discourse, world, question, or proof-state context; G_1 and G_2 are variable-bearing graph patterns; and a PLN intensional-implication truth value estimates how strongly the properties associated with G_1 support those associated with G_2 in C.

A semantic normal form such as FUSE-NF is important because intervention and chemistry require stable molecular shapes. Paraphrases, alternate syntactic analyses, and equivalent role expressions should be linked or consolidated enough that the system can intervene on a coherent semantic module rather than on one arbitrary surface phrase. The normalized graph may contain entities, events, semantic roles, modalities, temporal and causal links, source context, and uncertainty.

The rules are placed in contextual reaction chambers rather than used only as a static chain base. Graph fragments, question graphs, answer skeletons, causal motifs, scripts, and learned macros can act as molecules, catalysts, food, or products. Recurrent networks of transformations may become active autocatalytic sets (ACSs). External tasks reward useful question answering, compact explanation, hypothesis generation, proof progress, world-consistent story events, or improved action prediction.

Causal coding provides the accountability layer. A semantic rule can be plausible under PLN yet irrelevant to the task, or weakly plausible yet decisive because it opens a bridge. Paired replay, rule masking, chamber intervention, and ACS ablation estimate do-influence on answer quality, prediction error, hypothesis utility, narrative coherence, proof progress, or future ACS formation. Rules and loops with low marginal causal value pay rent or lose clarity. Useful loops become causal semantic ACSs: self-sustaining, rewarded, and intervention-relevant.

TECAN supplies the metabolism. A semantic rewrite additionally requires typed fuel, consumes that fuel, and emits a trace. A heritable semantic organism contains a quoted rule genome, a local graph state, a typed fuel vector, a chamber, and an event log. It may be archived, expressed, mutated, recombined, or reproduced only under explicit cost and evaluation. This prevents cheap self-reinforcing loops from dominating merely because they fire quickly.

Semantic chemistry is useful beyond natural language. Robotic scene and action graphs can learn affordance and repair loops. Biomedical pathway graphs can generate multi-mechanism research hypotheses. Mathematical proof graphs can discover lemma motifs and proof-state transformations. Software graphs can identify bug and patch ecologies. In the world-model architecture, semantic chemistry belongs mainly to the ambient loop: it explores reusable transformation networks, while PLN, evidence anchoring, causal replay, and action outcomes decide which networks deserve to influence live control.

A minimal experiment should use a constrained corpus and explicit tasks, not open-domain ambition. It should measure task performance, reward per compute and token type, number and transfer of

active ACSs, ablation impact, redundancy, cross-context commutators, replay reproducibility, and whether the discovered loops survive independent evaluation.

8.15 WILLIAM-on-MORK: Efficient Adaptive Compression for General Cognition

WILLIAM is a novel adaptive-compression-based AGI approach introduced by Arthur Franz, which is now being integrated into Hyperon in a fashion enabling very efficient execution on the MORK infrastructure.

WILLIAM operationalizes the principle that patterns worth retaining are often those that compress observed experience effectively. It functions as a cognitive feature detector that asks which compact explanation captures the current evidence. Moving WILLIAM into MORK's infrastructure changes it from a batch-oriented analysis tool into an online guide for symbolic reasoning and neural processing; release-specific latency and throughput remain benchmark questions.

The impact extends beyond pattern discovery. When WILLIAM identifies high-value structures, they become available to guide proof search, focus neural attention, and prioritize scarce computation. The compression score supplies a shared heuristic rather than requiring a separate hand-authored rule for every use case; whether it identifies task-relevant structure is evaluated against downstream outcomes.

8.15.1 Technical Implementation

MORK's trie nodes now carry additional instrumentation: local occurrence counts, subtree totals, compression-gain sums, and ranked lists of top-k children by various metrics. This seemingly simple addition enables WILLIAM to expose weighted iterators that return heavy subpatterns directly from any point in the graph–no global scans required.

The API surface remains minimal and intuitive, e.g.:

- iter_prefix_topk(prefix, k) returns the k most important patterns under a prefix
- iter_any_topk(k) finds globally significant patterns
- A validation API records actual compression gains when patterns are applied

These iterators serve multiple masters. PLN uses them to prioritize inference on high-value subgraphs. Backward chaining follows "heavy edges" that are likely to succeed. Schedulers allocate resources based on compression-adjusted priorities. The implementation handles concurrency through per-core write buffers for counts, with wait-free readers using snapshot/RCU techniques.

8.15.2 Neural Network Acceleration

WILLIAM displays a strong potential for elegant integration with transformers and predictive coding networks. Applied to the internals of a neural system, especially one doing local learning (with their greater propensity toward compositional representations), it has the potential to find "heavy-hitter features" and uses them to:

- Prioritize which attention heads and tokens receive computation
- Trigger on-demand refinement only where uncertainty justifies the cost

- Prune low-value frequency bands while preserving model capability

The weakness prior provides the theoretical foundation, explaining why simpler patterns generalize better. Rather than ad-hoc sparsification rules, we have a principled framework that applies uniformly across symbolic and neural domains.

8.16 Budgeted Pattern Mining with Dependency Towers and STLM

The newer pattern-mining design reframes surprisingness as a change in belief under additional inference budget[18]. For statement or pattern S, small budget B_0, larger budget B_1, and truth-value distance d, define

$$\Delta_{B_0 \to B_1}(S) = d\big(\mathrm{TV}(S, B_1), \mathrm{TV}(S, B_0)\big).$$

A pattern is interesting when thinking harder changes the system's estimate substantially. Classical independence-based surprisingness is a special case in which B_0 computes a factorized expectation and B_1 computes the full observed conjunction.

A Support Tensor Logic Machine (STLM) learns to predict this budget-delta from canonicalized logical templates, cheap endpoints, shallow traces, type signatures, cost proxies, and structural features. The miner expensive-evaluates only high-value or exploratory candidates, then adds the results to the training set. This turns pattern mining into learned inference control rather than blind enumeration.

When mining repeatedly over one knowledge base, dependency towers provide a compilation target. If relevant patterns expose at most k interface variables or holes, a k-ary tower stores canonical factors of arity at most k, content-addressed and indexed for bounded-width joins and heavy-first retrieval. It can collapse many equivalent fragments and make low-order evaluation nearly table-driven.

Projection of higher-arity relations into a low-arity tower can create ghost tuples when projected factors are rejoined. The architecture therefore distinguishes two valid modes. Tower-only mining finds patterns surprising relative to the best low-order reconstruction. Exact mining uses the tower for candidate generation and cheap features but verifies the expensive endpoint against the raw knowledge base. The final semantics remain exact even though search is accelerated.

STLM-guided search can reach structures that greedy minimum-support growth misses. It can glue small motifs into macros, propose multi-clause bundles, score large structured samples directly, preserve rare but high-margin branches, cluster analogical motif families, and make geodesic jumps in a tower-quotiented search space. In Hyperon, these mechanisms extend WILLIAM rather than replace it: WILLIAM supplies compression and heavy-hitter signals, while STLM predicts where additional inference will change the answer enough to justify its cost.

The same machinery can compile OmegaClaw Context Frames and world-model slices. Candidate memories, rules, episodes, or affordance templates can be evaluated cheaply, and the controller can spend deeper reasoning only on items expected to alter action prediction or strategic choice. TECAN can budget the expensive endpoint, while raw-KB verification and evidence closure keep compiled approximations from becoming unearned facts.

8.17 How These Advances Work Together

These recent cognitive-design additions are not independent improvements; they form a coherent enhancement to PRIMUS's capabilities:

1. **Weakness Based Control** selects regions needing attention based on importance and uncertainty
2. **WILLIAM** identifies compression-worthy patterns in those regions

3. **Fluid-Dynamic ECAN and Learning** route attention under an explicit transport objective toward regions predicted to be causally useful
4. **PLN** and **ActPC-Chem** refine understanding using both logical and chemical dynamics
5. **Schrödinger Bridge Learning** defines testable transport-based curricula from abstract to detailed models
6. **MetaMo/SubRep** evaluates which discovered options serve current goals
7. **TransWeave** determines what can be reused when facing new challenges
8. **PC layers** scheduled by WILLIAM and regularized by weakness theory process neural aspects

Throughout this flow, the same mathematical principles apply:

- **Weakness** governs all simplicity decisions: PLN scheduling priorities, WILLIAM's pattern selection, TransWeave's transfer filters, and PC network regularization all use the same quantale-based framework.
- **Geodesic control** guides planning through the canonical score $S_x(a)$ in Equation 30; forward reachability, backward usefulness, baseline, and cost use the same definitions in inference, evolutionary search, transfer studies, Schrödinger-bridge approximations, and motivational decisions.
- **Optimal transport** unifies dynamics: fluid-dynamic attention routing, Schrödinger bridge learning trajectories, and ActPC-Chem's gradient flows all leverage entropic transport principles.
- **TransWeave based knowledge transfer** spans cognitive components by viewing diverse cognitive algorithms effectively as approximate stochastic dynamic programming and guiding their interoperation accordingly
- **Content addressing** unifies storage: components–templates, scoped validation reports, chemical rules, transfer maps, and bridge potentials–can be CID-addressed MORK objects, making byte identity, deduplication, and integrity linkage efficient. Lookup and verification cost remains dependent on index, cache, payload, and availability conditions.

The integration of these advances in a common mathematical and conceptual framework enables capabilities that none could achieve alone:

- Patterns discovered by WILLIAM become chemical rules in ActPC-Chem
- Chemical dynamics discover structures that TransWeave can transfer to new domains
- Transferred solutions maintain SubRep validation reports, preserving the assumptions and tested scope recorded in those reports
- Attention flow patterns learned through fluid dynamics can be transferred via TransWeave to new contexts
- Schrödinger bridges provide curricula that guide both neural and evolutionary learning
- Weakness regularization supplies a shared simplicity and interference diagnostic intended to reduce brittleness as capability increases; its effect is established by ablation and stress testing
- MetaMo's motivation system coordinates all these processes toward coherent goals

8.18 The Path Forward

These new additions transform PRIMUS from a capable cognitive architecture supporting flexible cross-paradigm integration into something more ambitious: a system with a specific methodology and mandate for integrating the semantics and dynamics of different cognitive algorithms and the transfer of all sorts of knowledge between algorithms and tasks.

The significant aspect is not any single capability, but that the components are designed to share mathematical and representational foundations rather than being joined only through ad hoc interfaces. This provides evidence for the direction. The stronger proposition–that general intelligence emerges from these principles–remains a research thesis to be evaluated through integrated benchmarks, ablations, and externally reviewed formal claims.

9 Longstanding PRIMUS Components in the Current Architecture

While the newer additions to the PRIMUS framework are exciting, much of the cognitive heavy lifting is still done by stalwart components of the PRIMUS design which have been with us since OpenCog Classic (and in some cases the Novamente Cognition Engine and Webmind AI Engine before that). These venerable AI methods now operate with significant high-level control and low-level implementation extensions. They are integrated into the geodesic control framework for use with the compositional weakness, TransWeave and MetaMo approaches. They are also being implemented in MORK-customized ways for scalable in-RAM efficiency (a process known informally as "Morkification").

9.1 PLN (Probabilistic Logic Networks).

PLN is PRIMUS's uncertain reasoning layer: it treats knowledge as a typed metagraph of propositions and rules, and carries both *what* we believe and *how strongly* we believe it via simple, interpretable truth values (strength and confidence). Alongside its sophisticated uncertainty management, PLN's design has always emphasized two requirements–flexible "omni-ward" chaining (forward, backward, inward, outward) and programmable inference control–so the system can steer effort where it matters, not just produce long proofs. The current PLN approach keeps that spirit but streamlines the implementation by adding new options like guiding chaining geodetically, or implementing inference as message-passing on a factor-graph.

In this PLN formulation we can implement inference as "geodesic control": maintain forward factors from premises and backward factors from goals, then select steps using Equation 30. Evidence identity and duplicate accounting remain in the ledger rather than being inferred from the path geometry. Interleaving with neural critics or evolutionary search is governed by measured order defects under the relevant TransWeave maps.

To implement PLN natively on MORK, one can either use backward or forward chaining processes that traverse the MORK prefix trees at a low level (incorporating geodesic control heuristics at this level as well), or one can lay out a factor graph atop MORK's prefix trees and leverage this for a more decentralized and bottom-up form of control.

9.1.1 Context-Indexed Evidence Semantics in ωPLN

The most important conceptual change is that the persistent PLN state is no longer best viewed as one truth value per Atom, or as one global probability model. ωPLN stores context-indexed positive and negative evidence packets with provenance, relevance, assumptions, and ontology boundaries. A

simple truth value remains a useful interface, but it is projected from this richer state for a particular query or decision context.

Local Bayesian charts are constructed when a calculation needs a coherent probability space. The chart records its prior, factorization assumptions, language fragment, and universe of discourse; on export, prior mass is removed so that repeated chart construction does not create evidence. Cross-context evidence is merged only after an explicit translation into a covering context. A failed translation or a non-gluable family of charts is retained as a structured obstruction rather than forced into a misleading average.

Inference control and evidence accounting remain separate. Geodesic control may allocate high search mass to many paths, while provenance-aware idempotent fusion recognizes that those paths carry one observation. Reversible residual records make consequential revision and context translation auditable, and local commutator probes identify rule pairs whose order materially changes the result. Section 8.2 summarizes this architecture in more detail.

9.1.2 PLN Control via Quantale-Annotated Factor Graphs

"Classical" PLN tended to chain inference rules sequentially, which remains appropriate when a focused query touches a limited region. Even with geodesic guidance, however, sequential chaining can be inefficient for broad background consolidation over a large knowledge base. Factor graphs provide an alternative: truth values and evidence propagate as local messages through the graph, enabling parallel processing and explicit suspension and resumption of inference.

One key insight is keeping logical structure and uncertainty measures together through the message propagation process. Rather than maintaining separate systems for "what we know" and "how certain we are," every piece of information carries both aspects through the same computational flow.

The Technical Implementation The proposed encoding represents truth values (strength and confidence) as elements in a commutative quantale, supplying algebraic operations for uncertainty propagation. Pairing this with a logic-formula quantale creates a product construction that can carry structural and probabilistic information through the factor graph under the explicit mappings and coherence conditions stated for the implementation.

In practice, each proposition becomes a VariableAtom carrying truth-value payloads, while each inference-rule instantiation becomes a FactorAtom with a local potential function mapping input truth values to outputs. FactorVarLinks connect these elements, and MORK's PathMap structure is designed for low-latency neighbor lookup. Any latency result remains schema-, cache-, hardware-, concurrency-, and workload-dependent and should be reported with the software version, test hardware, data model, query mix, cache state, warm-up procedure, p50/p95/p99 distributions, failure rates, comparison baselines, acceptance thresholds, and raw results rather than described as instantaneous.

The orchestration layer handles operational requirements of distributed inference: damping to limit oscillation, termination detection, independence-error heuristics to reduce double-counting of evidence, and capsule logging for debugging and provenance.

9.1.3 Local Bayesian Islands and Background Inference

The combined-quantale factor graph should not be interpreted as one universal joint probability model. In ωPLN it is a strict control graph connecting local Bayesian islands, evidence packets, rule factors, revision factors, weakness and cost labels, and provenance. Each local chart may use

sum-product or another probabilistic method under explicit assumptions; its exported messages return to context-indexed evidence form.

This architecture is especially valuable for ambient background inference. Focused backward chaining remains preferable when a specific goal touches a small neighborhood. Factor-graph message passing is preferable when PRIMUS wants broad, incremental consolidation across a large knowledge base: surfacing likely affordances, maintaining dialogue context, propagating science relationships, or identifying low-confidence regions for targeted reasoning. MORK PathMap indexes make local neighbor updates efficient, while TECAN and weakness prioritize which messages and chart refinements deserve compute[21].

9.1.4 PLN Proofs as Procedural World-Generators

A proof DAG can also be compiled into an executable local world model. Intermediate proof Atoms become latent variables; deduction, abduction, conjunction, and revision become kernels or factors; truth values parameterize the kernels; and inference trails preserve provenance and dependency. Samples are reinserted as hypothetical witness structures and evaluated by forward PLN[20].

This gives PLN a constructive role in imagination. Instead of returning only "X has strength s and confidence c," it can return several explicit worlds in which X holds, partially holds, or fails for identifiable reasons. These worlds can generate predicted observations, candidate experiments, explanation structures, or plans. They remain quarantined from empirical evidence, and many samples from one proof count as one proof-derived source rather than many independent observations.

9.2 Evolutionary Program Search

MOSES is Hyperon/PRIMUS's automatic program learning methodology, bringing estimation of distribution algorithms together with evolutionary programming. It searches over spaces of small, human-readable programs–controllers, predicates, macros–that live directly in the Atomspace. It organizes the search into local "demes" (nearby template families), learns which parameters co-vary, and samples promising variants; the GEO-EVO upgrade adds two-ended guidance: search forward from what we have and backward from what we want, prioritizing candidates by the canonical reachability-and-usefulness-per-cost score rather than by unguided mutation alone. The hypothesis is faster discovery of reusable cognitive skills that plug back into reasoning and planning without being opaque; the speedup and solution quality must be measured against declared baselines.

MOSES maps directly into MORK when the evolved programs are represented as MORK subtrees. Program templates, factor-graph priors, and deme statistics are content-addressed Atoms. MORK can share byte-identical structures, index parameter neighborhoods, and stream changed-subgraph deltas. The resulting sampling and throughput benefits are workload-dependent engineering hypotheses to be measured against non-MORK baselines.

TransWeave enriches the picture by casting evolution as stochastic dynamic programming and supplying typed transfer operators between logic, evolution, and neural value learners ("do update, then map" $\approx$ "map, then update"). PLN subgoals can bias evolutionary variation; evolutionary regularities can feed back as new rules; and neural critics can provide amortized value estimates. Reordering is admitted only when the commutator gap in Equation 36 is within the declared budget.

9.2.1 GEO-EVO: Evolutionary Program Search

Deep integration of MOSES with TransWeave yields GEO-EVO–an enhancement that adds bidirectional guidance to the evolutionary picture: forward search from what we have and backward "teleological" factors from what we want, with candidates ranked by a declared reachability, usefulness, and cost model. This is more structured than unguided mutation because it exposes the program-space metric, endpoints, and selection objective. The optimal-transport interpretation supplies a research model for seeking low-cost paths through program space; global optimality, faster convergence, and transfer quality are empirical or theorem-specific claims rather than automatic consequences.

GEO-EVO's two-ended guidance maintains forward reachability factors (can we get there from here?) and backward usefulness factors (is it useful for our goals?). The system can expand search where the canonical score $S_x(a)$ is largest while enforcing a declared per-step cost band. The hypothesis is that this improves the balance between exploratory diversity and goal-directed progress; that benefit is established by comparative search experiments, not by the score definition alone.

9.3 ECAN (Economic Attention Network).

ECAN is Hyperon's primary, generic attention allocator. Other cognitive components may have their own internal attention allocation but there is still a need for higher-level attention allocation that helps guide attention across all subprocesses in the system. ECAN does this by spreading two "currencies"–short-term importance (STI) and long-term importance (LTI)–over the metagraph, forming an attentional focus that prioritizes computation and memory movement. The design is intentionally simple: importance rents, diffusion along links, and accumulation of durable relevance, tuned so the system neither fixates nor thrashes. That mechanism proved powerful in earlier OpenCog work and carried into Hyperon with better scaling and clearer knobs.

ECAN's importance-based selection can be implemented with high efficiency on MORK via embedding statistical records related to importance sampling directly on MORK nodes. And information-geometric (geodesic) methods have been used to accelerate ECAN since the OpenCog Classic days, which fits right into the TransWeave perspective. Fluid-dynamical methods may also be used to guide attentional flow on a more microscopic time-scale inbetween ECAN iterations.

TECAN deepens this picture by treating STI as a visible summary of typed liquidity rather than as the only spendable currency. Matching, deduction, abduction, causal replay, context bridging, provenance, compression, and mutation can receive separate budgets and shadow prices. Useful work earns evaluator-minted fuel; redundant or unsupported loops pay rent and may starve. This lets ECAN remain the architecture-wide attention interface while making the underlying resource semantics precise enough for OmegaClaw, PLN, semantic chemistry, and self-modification. See Section 8.5.

9.4 Pattern mining on general graphs.

Pattern mining is an indispensable low-level cognitive heuristic that finds reusable structures–templates that recur across contexts–and uses them to prime reasoning, suggest subgoals, and compress knowledge. The approach we have taken to this in Hyperon is incremental and stream-based: generate abstract templates, specialize, filter by support, expand conjunctions, and rank by "I-surprisingness"–but keep everything as nondeterministic streams until the final tally. That makes it natural to run continuously as knowledge drifts, and to reuse the results directly inside PLN and other cognitive methods.

To extend Pattern Mining into TransWeave, we can view mining as two-ended geodesic discovery between a data-driven prior over candidate patterns and a goal-driven posterior over useful patterns for

downstream tasks. Under this lens, the system tests whether mining steps and BD/bridge operators have bounded order defect on a named domain before interleaving them. Alignment and utility remain separately evaluated on the downstream task.

MORK implementation of pattern mining is especially direct when searched patterns are expressed as subtrees. In that case, matches are PathMap traversals; candidate templates are stored once and referenced many times; and counts or metrics are kept as capsule summaries at the nodes. Weakness-guided search can concentrate on high-priority subgraphs and expand when attention shifts, avoiding full global scans in the indexed cases for which the relevant patterns align with subtree structure. Throughput and scaling remain workload-dependent benchmark questions.

The STLM and dependency-tower approach adds a learned controller and a compiled substrate. Surprisingness becomes the change between a cheap and an expensive inference endpoint. STLM predicts which templates merit the expensive endpoint, and a k-ary tower caches canonical low-order factors for bounded-width search. When higher-arity atomic structure is projected into the tower, exact mode verifies final candidates against the raw Atomspace to eliminate ghost reconstructions. This makes it practical to preserve rare but structurally promising branches, glue motifs into larger candidates, and jump through pattern space without enumerating every intermediate template. Section 8.16 gives the broader picture.

9.5 Concept blending.

Concept blending is a primary heuristic for creativity – it creates novel ideas by merging properties from two or more source concepts so that the result shows useful emergent structure (think "sky-garden" = aerial platform + botanical garden). In Hyperon this is not a bag-of-words trick: blends are metagraph constructs that can be tested by the same evaluators that rate patterns, proofs, or programs, which makes blending a first-class, auditable operation rather than a one-off heuristic.

In a TransWeave setting, blending becomes a transport path in concept space: choose maps that minimize a declared effort objective while testing the constraints contributed by each source. Blending may be interleaved with pattern mining, inference, or program search only when the relevant typed maps and measured braid or order residuals remain within budget; consistency with the value geometry is then an evaluated condition, not an inherited label.

To support this process on MORK, sources, alignments, and resulting blends are content-addressed; byte-identical shared subparts can be deduplicated by content identity. Transport metadata–including contribution and cost records–is stored alongside the blend so later transfers can reuse named alignments, subject to availability and target-domain revalidation.

10 Neural-Symbolic Synergy in Hyperon

Large transformer networks have demonstrated remarkable capabilities in language, vision, code, multimodal generation, and broad statistical generalization. At the same time, an LLM-only architecture remains incomplete as a route to robust AGI. It does not natively provide a persistent typed world model, explicit motive structures, calibrated uncertain inference, durable procedural memory, principled attention economics, transparent self-modification, or a clean way for multiple learning and reasoning mechanisms to share intermediate state.

Hyperon's goal is not to reject transformer networks but to place them in a broader cognitive system. Neural models can provide high-bandwidth perception, generation, and implicit pattern completion. Atomspaces provide structured memory and a common control plane. PLN supplies

uncertain reasoning and proof-like provenance. WILLIAM and pattern mining identify reusable structure. Predictive coding supports local, recurrent, error-driven adaptation. MOSES and related methods search program space. ECAN, MetaMo, SubRep, and geodesic control determine what deserves computation and why. OmegaClaw turns these elements into deployable Module Spaces governed through Context Frames and attention loops. OmegaSelf records the evidence for claims about their current capabilities, scores system-level predictions, and keeps neural confidence or low latent error from automatically becoming action authority; an exact-input policy gate remains the separate authorization mechanism.

The world-model architecture gives these neural interfaces a concrete cognitive role. Symbolic heads are not only retrieval augmenters: they participate in Γ and Λ, grounding evidence into hypotheses and lifting active entities, rules, norms, HMH episodes, and options into dense context and gates. Predictive-coding layers implement portions of S_{dyn}, while HMH retrieval supplies structured action and episode priors. The bridge should therefore be evaluated not only on language benchmarks but on action-conditioned prediction, identity persistence, re-perception, staleness handling, counterfactual discrimination, and continual learning across world contexts[15].

10.1 Three Paths to Neural Integration

The architecture distinguishes three integration paths. They form a spectrum of commitment rather than mutually exclusive alternatives.

1. **External Neural Spaces.** A model remains in its native serving stack. Hyperon invokes it through a typed adapter and records prompts, outputs, embeddings, confidence proxies, provenance, and evaluations as Atoms. This path works even when weights and internal activations are unavailable.

2. **Predictive-Coding and Symbolic-Head Bridge.** An open-source or otherwise instrumentable transformer remains the neural kernel, but selected hidden states are connected to symbolic memories, predictive-coding nodes, sparse residual adapters, and OmegaClaw control. Most base weights can remain frozen. This path seeks strong neurosymbolic interaction with moderate engineering risk.

3. **Native Neural Computation in Atomspace.** Neural state and operations are represented inside Hyperon, for example using QuantiMORK's multiresolution tensor structures and local predictive-coding updates. This path offers the deepest sharing of memory, scheduling, and symbolic intervention, but requires more specialized neural architectures and runtime engineering.

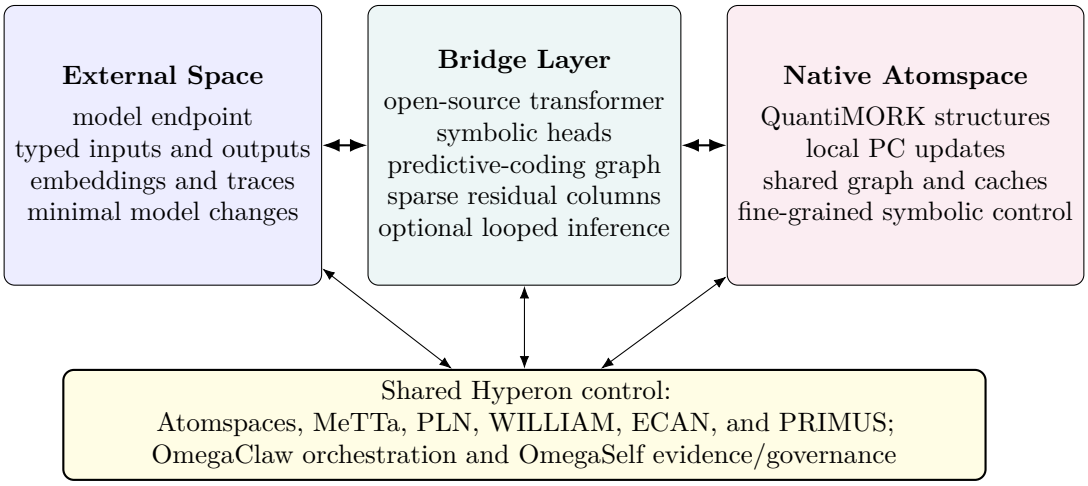

Figure 9: Three neural integration modes. A deployment can use all three at once and move capabilities between them as implementations mature.

The middle path is strategically important. External APIs can be useful immediately but provide limited access to internal cognition. Fully native neural computation is a major research and engineering programme. A bridge on top of open-source transformers can connect current high-capacity models to Hyperon's structured memory and learning loops while preserving the option to migrate selected functions deeper into Atomspace later.

10.2 The Transformer as a Virtual Hard Kernel

A practical starting point is a pretrained open-source decoder or multimodal transformer F_0. The base can initially be frozen or changed only through carefully bounded procedures. Freezing is not a philosophical commitment; it is an engineering tactic. It preserves a known capability baseline while making the causal effects of new symbolic, predictive-coding, and residual mechanisms easier to measure.

The transformer is treated as a virtual hard kernel: a dense, high-capacity substrate that already contains linguistic, perceptual, and procedural regularities. Hyperon adds a softer, more explicit cognitive layer around it. This layer can:

- observe selected residual-stream states, attention outputs, logits, and uncertainty signals;
- retrieve structured knowledge, rules, episodes, task frames, and constraints from Atomspace;
- generate symbolic hypotheses or graph fragments from neural state;
- run recurrent prediction-error correction without retraining the whole transformer;
- apply sparse, auditable residual corrections to known weaknesses;
- allocate extra inference-time computation only when predicted benefit exceeds cost;
- record the whole process in a Context Frame so other cognitive modules can inspect and challenge it.

The bridge may attach at one layer, a small set of layers, or a contiguous middle window. Early experiments should prefer conservative insertion points and compare against the unmodified model on

a broad anchor set. A mechanism that improves one benchmark while damaging calibration, fluency, or general behavior is not an improvement to the system as a whole.

10.3 Symbolic Heads: Structured Read and Write Interfaces

A symbolic head is a trainable or partly rule-based interface between transformer hidden states and Atomspace. It is not merely retrieval-augmented generation at the prompt boundary. It can operate at selected internal layers, can return typed graph structures rather than plain text, and can write candidates back to a shared memory that other modules reason over.

10.3.1 Reading from Atomspace

For a hidden state h_i at token or patch position i, a read head produces a query $q_i = W_q h_i$ or a structured query generated by a small controller. It retrieves a small set of relevant Atomspace objects:

- semantic and episodic memories;
- mined templates and recurring subgraphs;
- task schemas, dialogue frames, proof motifs, or procedural options;
- constraints, invariants, capability policies, and current goals;
- PLN conclusions and their truth values;
- Context Frame state such as active hypotheses and unresolved evidence gaps.

Retrieved objects can be encoded into vectors and injected through cross-attention, low-rank residuals, gated adapters, or predictive-coding priors. The structured object identity is retained in the Context Frame, so a later audit can distinguish which Atom influenced an output from which vector happened to be similar.

10.3.2 Writing to Atomspace

A write head maps neural state to candidate symbolic objects. It may propose an entity, relation, event, rule, causal hypothesis, code fragment, plan step, or confidence estimate. These are candidates, not automatically accepted truths. OmegaSelf treats an LLM or symbolic-head assertion as testimony until its source type, evidence, and validation are clear. A write policy can require:

- type validation and schema checks;
- deduplication against existing atoms;
- evidence links to the source tokens, images, tool results, or hidden-state region;
- PLN or external verification where appropriate;
- truth-value assignment and calibration;
- permission checks for sensitive or persistent regions;
- a promotion path from ephemeral working memory to shared or on-chain storage.

This read-write symmetry is central. Symbolic memory should not only tell the model what to say; neural processing should continuously propose structure that the broader cognitive system can test, refine, combine, or reject.

10.3.3 Where Symbolic Content Comes From

WILLIAM and Hyperon pattern mining can create symbolic-head content by finding frequent, compressive, or surprising structures in documents, interaction histories, program traces, proofs, and neural activation summaries. A generic pipeline is:

$$\begin{aligned}&\text{generate abstractions} \longrightarrow \text{specialize} \longrightarrow \text{filter by support}\\ &\quad\longrightarrow \text{compose conjunctions} \longrightarrow \text{rank by information value and weakness.}\end{aligned}$$

The resulting templates need not form a fixed library. MeTTa stream operations can retrieve or construct patterns lazily. A symbolic head can ask for structures matching the current task, motive, domain, or uncertainty profile. Templates that repeatedly improve performance can be consolidated; those with low causal value can decay or be pruned.

10.4 Predictive Coding as the Adaptive Bridge

Symbolic heads provide structured interfaces, but they do not by themselves create a coherent recurrent learning dynamic. Predictive coding supplies such a dynamic. In a predictive-coding network, nodes maintain states and predictions of neighboring states. Inference updates latent states to reduce local prediction errors; learning updates the parameters that generate predictions. Top-down expectations and bottom-up evidence meet through iterative error correction rather than through a single feedforward pass.

For a bridge graph with states z_i, predictions $\widehat{z}_i$, precisions Λ_i, and symbolic or task constraints Φ_{sym}, a schematic energy is

$$E(z,a) = \sum_i \|z_i - \widehat{z}_i\|^2_{\Lambda_i} + \lambda_{\text{sym}}\Phi_{\text{sym}}(z,a) + \lambda_{\text{anchor}} D(p_{\text{bridge}}, p_0), \tag{41}$$

where a denotes the relevant Atomspace structures, p_0 is the frozen base behavior, and D penalizes unacceptable drift on anchor data. The symbolic term can encode logical constraints, task expectations, retrieved templates, or consistency with a Context Frame. The precisions can be influenced by PLN truth values, learned uncertainty, or attention.

A bridge can include several node types:

- observations of transformer residual states or logits;
- learned predictors between selected layers or timescales;
- symbolic-memory embeddings and constraint nodes;
- associative memories such as Hopfield-style components;
- task, motive, and Context Frame nodes;
- error nodes whose magnitude and persistence become explicit attention signals;

- action or decoding nodes that feed corrected states back to the transformer.

Inference can occur for a small fixed number of iterations, until an error threshold is met, or under a budget chosen by OmegaClaw. Learning may be local to the bridge, sparse residual adapters, or symbolic-head projections. The base transformer can remain frozen until evidence justifies a broader change.

10.4.1 Why Predictive Coding Fits Hyperon

Predictive coding aligns with Hyperon in several ways.

First, local prediction errors are explicit objects that can be stored, attended to, mined, and explained. A persistent high error can trigger PLN reasoning, a tool call, or a new subgoal rather than silently changing a weight.

Second, recurrent inference naturally accommodates top-down symbolic expectations and bottom-up neural evidence. A proof constraint, a task schema, or a sensor observation can enter the same energy-minimization process with different precisions.

Third, local updates are compatible with asynchronous and distributed execution. Different regions can refine at different rates, and OmegaClaw can allocate compute to high-value errors.

Fourth, predictive-coding graphs can include heterogeneous nodes rather than only copies of one neural layer type. This makes them a natural bridge between transformer blocks, structured memories, and cognitive-control variables.

10.4.2 Two Prediction Levels: Latent State and Agent Behavior

Predictive coding in the neural bridge and prediction in OmegaSelf are related but not identical. The bridge predicts hidden states, observations, symbolic embeddings, or neighboring node activity and uses local errors to refine a latent trajectory. OmegaSelf predicts the behavior of the running agent as an operational system: whether the selected module will succeed, how long it will take, what it will cost, which failure mode is likely, whether a symbolic write will survive verification, and whether an architectural change will preserve commitments and continuity.

The two levels should exchange evidence without collapsing their authority. Persistent latent error, loop drift, route thrash, or symbolic inconsistency can become typed OmegaSelf observations and can lower a contextual capability estimate. Conversely, OmegaSelf can ask OmegaClaw to allocate more predictive-coding iterations, choose a different loop window, retrieve stronger symbolic constraints, route to another module, or require an external probe. A low predictive-coding energy is not proof that an action is permitted or that the system-level answer is correct; a high operational confidence is not a substitute for stable latent dynamics. The separation makes both forms of prediction more measurable.

10.5 FabricPC as One Concrete Engineering Scaffold

FabricPC is an open-source JAX library for defining and training predictive-coding networks using three core abstractions: nodes, edges, and updates[8]. It supports feedforward, recurrent, skip, and cyclic graphs; heterogeneous components such as linear nodes, transformer blocks, and Storkey-Hopfield associative memory; GPU and multi-GPU execution; and the ability to train the same graph topology with predictive coding or backpropagation for controlled comparisons.

These properties make FabricPC a plausible scaffold for early bridge experiments:

- a frozen transformer block can be wrapped as one or more graph nodes;
- Atomspace retrieval and symbolic constraints can be represented by custom nodes;
- recurrent error correction can operate across neural and symbolic states;
- local automatic differentiation can implement node-level derivatives;
- the same topology can be compared under predictive coding and backpropagation, separating architectural gains from training-algorithm gains;
- JAX provides a route to accelerator execution without requiring the entire Hyperon runtime to become a tensor framework.

FabricPC is not a mandatory dependency of Hyperon and its APIs may evolve. The deeper architectural requirement is a graph-based predictive-coding layer with typed connections, heterogeneous nodes, explicit local errors, and hooks into Atomspace and OmegaClaw. FabricPC is valuable because it makes that requirement experimentally accessible now.

10.6 Sparse Columnar Residual Adaptation

A predictive-coding bridge still needs a mechanism for learning narrow corrections without entangling the whole base model. Sparse columnar residual adaptation is one candidate. The basic idea is to attach many small rank-one adapter atoms to a frozen model. For a hidden state h and atom g, an atomic residual has the form

$$A_g(h) = s_g b_g(a_g^\top h), \tag{42}$$

where a_g is a read direction, b_g is a write direction, and s_g is a scale. Atoms are organized into sparse virtual columns. A router selects only a few columns for a given input, and the active columns combine their atoms into a residual correction.

This approach is attractive for Hyperon because the elements are small enough to be named, typed, grouped, ablated, and causally audited. A column can be associated with a weakness, domain, concept, task, or failure pattern. Its activation is not treated as proof that it matters; sampled exact ablations and learned information-geometric surrogates estimate its causal effect. WILLIAM can mine recurring activation and error structures, while weakness-based objectives can penalize redundant atoms, off-support gradient leakage, excessive cross-column coupling, and selector regret.

The HiBaCaML perspective generalizes this architecture into two probabilistic levels: a sparse upper-level controller selects, reuses, diversifies, or extends structurally restricted component learners, while each active component maintains an internal probabilistic organization that separates reusable causal structure from task-local residue [9]. The current columnar-Bayesian approach can be understood as one instance of that broader family. This matters because it reframes the goal from inventing one clever adapter to identifying environmental conditions under which sparse reusable causal modules should outperform undifferentiated continual learning.

10.6.1 Radial Roles for Columns

Columns need not all participate in the same way. A useful organization distinguishes:

- **inner or shared columns** representing stable, reusable abstractions;

- **middle columns** stabilizing recurrent dynamics or handling semi-general corrections;
- **outer columns** representing narrow task-specific residue.

This radial interpretation becomes especially useful when the transformer is looped. Shared columns may be safe to apply repeatedly. Stabilizing columns can improve the geometry of the loop. Task-specific columns are often safer as one-shot corrections after the recurrent computation is complete.

10.7 Optional Looped Inference-Time Computation

Training-free looped transformers suggest that a frozen transformer may contain latent computation that can be exposed by reapplying a narrow middle-depth layer window [11]. Let a contiguous window be

$$g = L_b \circ L_{b-1} \circ \cdots \circ L_a.$$

Naively applying g repeatedly can push hidden states far beyond the endpoint the later layers were trained to consume. A safer interpretation treats $g(h) - h$ as a residual vector field and subdivides approximately the same total step. A damped loop uses

$$h_{k+1} = h_k + \alpha_k(g(h_k) - h_k), \qquad \sum_k \alpha_k \approx 1. \tag{43}$$

Looped columnar residual adaptation combines this extra latent computation with sparse columns[12]. If U_{S_0} is the residual field of the active loopable columns, the inside-loop update becomes

$$h_{k+1} = h_k + \alpha_k \left(g(h_k) - h_k + U_{S_0}(h_k)\right). \tag{44}$$

The active support is selected once before the loop,

$$S_0 = \mathrm{TopK}_a\, \rho_a(h_0), \qquad S(h_k) = S_0 \text{ for } k = 0, \ldots, K-1, \tag{45}$$

rather than recomputed at every iteration. This pinned-support rule prevents the router and the recurrence from fighting each other. Without it, small state changes can cause column thrashing: each loop step selects a different intervention, turning iterative refinement into an unstable sequence of corrections.

Several variants should be tested in increasing order of risk:

1. **Loop then adapt:** run a conservative loop wrapper, then apply symbolic heads and residual columns once.
2. **Controller-only columns:** use sparse columns to choose the loop window, number of iterations, step sizes, caching policy, or bypass decision without writing to the hidden state.
3. **Pinned columns inside the loop:** allow stable shared or stabilizing columns to participate in recurrent refinement.
4. **Radial loop shells:** loop shared inner columns, apply narrow outer columns only after the loop, and let OmegaClaw select among policies according to task and budget.

The point is not that every input should receive more transformer computation. The controller should estimate expected gain per unit latency and bypass the loop when extra computation is unlikely to help.

10.8 A Schematic Bridge Model

The combined architecture can be summarized schematically as

$$F_{\text{bridge}}(x; \mathcal{A}, C) = \text{Post}(\text{PC}(F_0^{\pi}(x), H_{\text{sym}}(x, \mathcal{A}), A_{\text{col}}(x), C)) \,, \tag{46}$$

where:

- F_0^{π} is the frozen transformer under an optional loop policy π;
- $\mathcal{A}$ is the relevant Atomspace state;
- H_{sym} denotes symbolic read and write heads;
- A_{col} denotes sparse columnar residual corrections;
- C denotes Context Frame goals, hypotheses, evidence, and constraints;
- PC denotes recurrent predictive-coding inference over the connected neural and symbolic graph;
- Post denotes the remaining model layers, decoding, or action interface.

This equation is intentionally schematic. Symbolic heads may enter at several layers; predictive coding may correct hidden states, logits, memory embeddings, or action plans; columns may act inside or outside the loop; and the output may be text, code, a graph, a motor command, or a proposal sent to another module. The architectural invariant is that the bridge's state and causal traces remain addressable by Hyperon.

10.9 OmegaClaw, OmegaSelf, and PRIMUS Control of the Bridge

The bridge should not become a self-contained mini-AGI disconnected from the rest of the system. OmegaClaw, OmegaSelf, and PRIMUS control it at several levels. OmegaClaw orchestrates the components; PRIMUS supplies the task, attention, learning, and motive logic; OmegaSelf calibrates contextual capability, records the evidence and predictions, and governs consequential proposals.

10.9.1 Capability and Module Selection

The transformer, symbolic head, PC inference engine, column router, and loop controller may be exposed as one composite Module Space or several cooperating Module Spaces. The registry records which versions are available, their latency and memory cost, their domain strengths, and whether they support local execution, privacy, or accelerator hardware. OmegaSelf keeps declared availability separate from current reachability, authorization, and evidence-grounded effectiveness for the exact task and rubric.

10.9.2 Context and Goal Conditioning

A Context Frame provides the active goal, hypotheses, unresolved questions, retrieved precedents, safety constraints, and budget. Symbolic heads retrieve relative to this structured context, not merely

the last few thousand tokens. Predictive-coding precisions can be adjusted according to goal relevance and truth values. A high-confidence formal constraint should not be treated like a weak conversational hint.

10.9.3 Attention and Compute Allocation

The fast attention loop can schedule PC iterations, choose which error nodes to refine, and prioritize symbolic queries. The slow loop can decide that the current bridge is stuck, route the task to PLN or a tool, request a different model, change the loop policy, or ask a human. WILLIAM can mine execution traces to learn which classes of tasks benefit from which bridge configurations.

10.9.4 Promotion of Learned Structure

A column, symbolic template, or PC connection that appears useful enters a promotion pipeline. It is benchmarked, causally audited, tested for interference, run in shadow mode, and associated with a scope. Shared abstractions may be promoted toward the core; task-local residue remains peripheral. The same shadow, elevate, monitor, and rollback logic used for whole PRIMUS components applies at this smaller scale. Promotion is a governed transport: its proposal, benchmark evidence, prediction, policy decision, pre/post signatures, and rollback receipt become part of the OmegaSelf continuity record.

10.10 Weakness, Commutativity, and Dynamical Stability

A bridge introduces multiple update streams: neural prediction errors, symbolic retrieval, column learning, router learning, loop control, and task-level feedback. Without explicit coordination, improvements in one stream can damage another. Hyperon's weakness and TransWeave ideas provide a common language for these interactions.

For columnar adaptation, useful weakness terms include:

- atom weakness or poor causal specificity;
- redundancy within a column;
- leakage of gradients or effects outside the selected support;
- cross-column coupling;
- router or selector regret;
- semantic or categorical incoherence;
- continual-learning interference.

Looped systems add dynamic terms:

- endpoint drift from the region expected by downstream layers;
- non-contraction, measured by growing successive increments;
- route thrashing;
- cache inconsistency during autoregressive decoding;
- amplification of a weak column through repeated application.

Predictive-coding systems add commutator questions. If two local update processes X_i and X_j represent, for example, lateral correction and top-down correction, then a large commutator

$$\|[X_i, X_j]\| = \|X_i X_j - X_j X_i\|$$

indicates that update order strongly changes the result. Soft penalties on such order effects can encourage more compositional and asynchronous learning. The goal is not perfect commutativity, which may be neither possible nor desirable, but bounded order sensitivity that can be measured and managed.

A practical training objective can combine task performance with stability and audit terms:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{KL}} D_{\text{KL}}(p_0 \| p_{\text{bridge}}) + \lambda_{\text{sparse}} \mathcal{L}_{\text{sparse}} + \lambda_{\text{weak}} \mathcal{W}_{\text{col}} \tag{47}$$

$$+ \lambda_{\text{drift}} \epsilon_{\text{drift}} + \lambda_{\text{contract}} \epsilon_{\text{noncontract}} + \lambda_{\text{comm}} \mathcal{L}_{\text{comm}} + \lambda_{\text{audit}} \mathcal{L}_{\text{audit}}. \tag{48}$$

The weights and exact terms are empirical questions. What matters is that the architecture makes interference, drift, causal contribution, and calibration explicit optimization and evaluation targets rather than hoping that aggregate task loss will control them indirectly.

10.11 Causal and Trajectory Audits

Activation is not explanation. A symbolic template, column, or PC node may be highly active while contributing little, or may matter only because it changes a later recurrent trajectory. Audits should therefore be interventional.

For sampled atoms, columns, templates, or connections, the system can perform exact ablations and record:

- task loss and output changes;
- endpoint hidden-state perturbation;
- changes in loop drift and contraction;
- changes in PC energy and error distribution;
- downstream symbolic hypotheses gained or lost;
- effects on calibration and anchor behavior;
- effects on old tasks after continual learning.

Exact ablation of every component is too expensive. Information-geometric or learned interventional surrogates can estimate causal impact at scale, with periodic exact ablations used to recalibrate them. Looped computation naturally yields a short trajectory $h_0, \ldots, h_K$, which should be stored as an audit object or compressed causal fingerprint rather than discarded. A component's role can then be described by how it changes the whole trajectory, not only the final activation.

10.12 Experimental Programme

A disciplined programme should proceed from conservative baselines to deeper integration.

10.12.1 Stage 0: Establish the Base and Instrumentation

Choose one or more 1B–7B open-source models. Establish reproducible base scores, calibration, perplexity, latency, memory, and generation quality. Expose selected hidden states and define anchor datasets representing behavior that should not regress.

10.12.2 Stage 1: Symbolic-Head Read-Only Baseline

Add read-only Atomspace retrieval at one or a few layers. Compare prompt-boundary RAG, internal symbolic heads, and no retrieval. Measure task performance, calibration, retrieval relevance, provenance quality, and latency. The first goal is not continual learning but a clean causal demonstration that internal structured retrieval adds value.

10.12.3 Stage 2: Predictive-Coding Overlay

Build a small PC graph around selected transformer states, symbolic embeddings, and task nodes. Compare predictive-coding and backpropagation training on the same topology where possible. Test fixed versus adaptive inference steps and whether persistent errors provide useful signals to OmegaClaw.

10.12.4 Stage 3: Post-Transformer Sparse Columns

Train sparse rank-one columns as one-shot residual repairs while keeping the base fixed. Audit exact causal effects, support diversity, off-support leakage, cross-column coupling, and continual-learning retention.

10.12.5 Stage 4: Controlled Looped Inference

Reproduce a safe looped-transformer baseline: narrow versus wide windows, damped versus naive loops, block versus layer mode, fixed versus controller-selected policies. Evaluate accuracy per unit latency and require open-ended generation and calibration tests, not only multiple-choice accuracy.

10.12.6 Stage 5: Loop-Internal Columns and Controller Columns

Compare dynamic and pinned support. Test shared inner columns inside the loop and narrow outer columns after the loop. Train controller-only columns to choose when and how to loop. Measure route thrash, endpoint drift, contraction, anchor KL, and causal amplification.

10.12.7 Stage 6: Closed Symbolic Read-Write Loop

Allow the model to propose Atomspace writes subject to validation and promotion. Test whether mined templates improve future tasks, whether false structures are caught, and whether provenance remains intact across repeated read-write cycles.

10.12.8 Stage 7: OmegaClaw and PRIMUS Integration

Expose all cells as Module Spaces, record their operations in Context Frames, let the Attention Metaprotocol allocate compute, and compare fixed orchestration with learned routing. Run shadow promotion and rollback. Evaluate whole-task outcomes, not only model benchmarks.

10.13 Minimum Ablation Matrix and Metrics

At minimum, experiments should separate the contribution of each mechanism.

Cell	Loop	Columns	Symbolic / PC layer	Purpose
A	No	No	No	Frozen base reference.
B	Yes	No	No	Isolate inference-time looping.
C	No	Late	No	Isolate sparse residual adaptation.
D	No	No	Read-only symbolic head	Isolate structured internal retrieval.
E	No	No	Symbolic head plus PC	Isolate recurrent neural-symbolic correction.
F	Yes	Late	Symbolic head plus PC	Conservative full bridge.
G	Yes	Inside loop, pinned	Symbolic head plus PC	Higher-risk recurrent adaptation.
H	Policy only	Controller columns	Symbolic head plus PC	Adaptive compute without hidden-state column writes.

Table 8: A minimum bridge ablation matrix.

Metrics should include:

- task accuracy, loss, calibration, perplexity, and open-ended generation quality;
- KL or behavioral drift on anchor data;
- retrieval precision, provenance completeness, and symbolic-write validation rate;
- PC energy reduction, residual distribution, convergence speed, and precision calibration;
- loop endpoint drift, successive-increment ratios, route-thrash count, and cache consistency;
- exact sampled ablation impact and surrogate-audit error;
- off-support gradient leakage and cross-column curvature coupling;
- old-task retention, mean forgetting, and transfer to related tasks;
- latency, memory, energy, and accuracy per unit compute;
- task-level benefit when routed through OmegaClaw, including fallback and rollback frequency.

10.14 Expected Failure Modes

The bridge introduces several predictable risks.

Decorative symbolism. Symbolic heads may retrieve plausible structures that do not causally improve the result. Exact ablation and provenance-aware evaluation are the remedy.

False symbolic writes. Neural proposals may pollute shared memory. Writes need schema checks, evidence links, truth values, scoped persistence, and promotion gates.

Predictive-coding collapse or oscillation. Poor precisions or incompatible local updates can prevent convergence. Energy traces, commutator penalties, and bounded iterations make this visible.

Loop-column amplification. A harmless one-shot adapter can become harmful when applied repeatedly. Shared versus task-local roles, pinned support, drift penalties, and controller-only baselines are essential.

False loop gains. Extra computation may improve narrow benchmarks while hurting fluency, calibration, or latency. Broad evaluation and accuracy-per-compute are required.

Router thrash. Recomputing sparse support inside a recurrent loop can cause a sequence of changing interventions. Pinned support is the conservative default.

Over-specialization. Outer columns or task-specific templates may fit one domain and interfere elsewhere. Radial organization, scope annotations, and continual-learning tests help contain the damage.

Audit mismatch. Static component signatures can become invalid in a recurrent system. Audits must include PC and loop trajectories.

Control-fabric overreach. An orchestration policy may learn to call expensive or opaque modules unnecessarily. Context Frame budgets, module economics, and human or PRIMUS slow-loop review provide counterpressure.

10.15 Example Workflows

10.15.1 Workflow A: External Model with Structured Augmentation

A closed or remote model is invoked as a Neural Space. OmegaClaw places the task in a Context Frame, retrieves relevant Atomspace structures at the prompt boundary, sends the request, and routes the result to PLN or a verifier. This is the lowest-integration baseline and remains useful when no open model is competitive.

10.15.2 Workflow B: Open-Source Transformer Bridge

An open model runs locally. Symbolic heads retrieve task frames and evidence at selected layers. A small predictive-coding graph reconciles the hidden state with symbolic and Context Frame expectations. Sparse columns correct known domain weaknesses. A loop controller allocates extra mid-stack computation only for hard cases. The bridge returns both neural output and a structured trace of retrieved atoms, PC errors, active columns, loop policy, and confidence.

10.15.3 Workflow C: Native QuantiMORK Cognition

For a domain where multiresolution, sparse, local learning is central, neural state lives in MORK-backed structures. Weighted Atom Sweeps identify active regions. ByteFlow or related mechanisms map them to accelerator-friendly kernels. Predictive-coding updates flow locally. WILLIAM promotes recurring structures into symbolic templates and PLN rules. MeTTa orchestrates refinement and consolidation. This path maximizes integration but should be pursued where its structural advantages justify the specialized runtime.

10.15.4 Workflow D: Hybrid Cognitive Task under OmegaClaw

A single task may use all three modes. A local bridge model parses a scientific paper and writes candidate causal relations. PLN checks the relations against a biomedical Atomspace. An external frontier model is called only for ambiguous language. A native PC network analyzes a high-dimensional

sensor stream. The task planner runs these subtasks in parallel, the Attention Metaprotocol shifts budget toward unresolved evidence, and the Context Frame records the combined result. A completed reasoning trace may be committed to ASI:Chain if cross-organization verification is needed.

10.16 Inside Atomspace: The QuantiMORK Direction

The bridge is not a replacement for the more radical QuantiMORK direction. Putting neural state inside Atomspace can eliminate remaining serialization boundaries and allow symbolic rules to intervene at the same granularity as neural operations. QuantiMORK proposes to encode tensors through multiresolution structures, for example wavelet coefficients keyed by level, band, and position. The hierarchy can support selective refinement, progressive processing, and shared substructure.

Ground operations can perform dense kernels on selected subtrees while MeTTa rules schedule forward inference, local prediction-error correction, consolidation, and symbolic interaction. Computation can focus on high-error or high-attention regions. Different regions can update asynchronously. WILLIAM can track compression and identify heavy hitters, while weakness-guided pruning removes components whose causal contribution is small.

This direction remains a substantial research programme. Its importance is that Hyperon does not make today's transformer tensor layout an eternal architectural commitment. The bridge extracts value from open-source transformer ecosystems now; native Neural Spaces explore architectures better aligned with Atomspace, local learning, sparse scheduling, and transparent cognition in the longer term.

10.17 The Deeper Significance

The historical neural-symbolic divide has often been reinforced by software boundaries. Neural systems live in tensor runtimes, symbolic systems in graph or logic engines, and they exchange strings or vectors at a few endpoints. Hyperon treats this separation as a contingent engineering choice rather than a fundamental law.

External Spaces provide immediate access to strong models. Symbolic heads create typed read and write paths. Predictive coding makes the interface recurrent and locally adaptive. Sparse columns provide modular specialization and causal audit. Looped inference allocates additional computation without automatically retraining the base. QuantiMORK seeks deeper substrate unity. OmegaClaw and PRIMUS decide how these pieces serve goals, attention, learning, and governance.

The result is not "a transformer plus a knowledge graph" and not "a symbolic system that occasionally calls an LLM." It is a layered migration path toward cognition in which neural states, symbolic structures, hypotheses, errors, motives, module choices, and edits can become parts of one inspectable process. That is the level of integration needed for cognitive synergy to be more than a slogan.

11 Planning for the AGI → ASI Transition: Stability, Reflection, and Safe Self-Modification

11.1 The Challenge of Ongoing Radical Self-Improvement

Creating a capable AGI is one milestone in a longer engineering programme. A central part of the AGI challenge–and opportunity–is to build a generally intelligent system that can improve selected

components without abrupt or unobserved loss of purpose, policy compliance, or valued behavior. This is a practical requirement for any system expected to upgrade capabilities, refine algorithms, and scale processing over time. The crucial question is how such change can proceed while alignment-relevant properties, human authority, and core objectives remain explicit, tested, and governable.

Traditional software can be upgraded by external developers who understand its purpose. But an AGI that truly deserves the name must be able to modify itself, and those modifications must preserve the properties we care about – or at least, they must do so initially, and deviate from this course only after an appropriate level of care and deep reflection. This requires three interconnected capabilities:

1. **Goal preservation** - A way to represent and maintain objectives that survive through system upgrades
2. **Composition discipline** - Typed interfaces, residual budgets, and scoped validation that test whether modules continue to work together predictably as they evolve
3. **Controlled modification** - A procedure for self-improvement that includes validation, testing, and rollback

11.1.1 The Technical Framework

We address these challenges in the current Hyperon design through a combination of mathematical frameworks and engineering practices. Goal stability can be modeled as a multi-level fixed-point or invariant-region problem. Formal existence, uniqueness, or iteration-convergence conclusions are available only in an instantiated model for which the selected theorem's hypotheses are verified. Global regulators–geodesic effort and weakness-based simplicity–supply shared measurements and biases across modifications; they do not automatically prove safety. The self-modification pipeline treats changes as typed transformations with scoped admission reports, simulation testing, and capability-controlled deployment.

11.2 Goal Stability as a Tractable Math Problem

Goals are not monolithic–they are layered structures where higher levels constrain how lower levels can evolve. Consider human values: we have immediate goals (finish this task), persistent objectives (maintain health), core values (preserve human dignity), and meta-values about how values themselves should change (remain open to growth while preserving identity). The relation between the goals on different levels is not purely top-down nor bottom-up but is more like the relationship between the layers in a predictive-coding neural network: ongoing iterated inter-modification leading to holistic self-organization of the network across all levels.

An AGI needs the same kind of hierarchical goal structure. Each level acts as an invariant that constrains acceptable changes to the level below. This creates a "stack" of purposes where evolution is possible but bounded.

11.2.1 The Mathematics of Stability

We formalize this through invariant hierarchies. Goals become a family $I = \{I_1, I_2, \ldots\}$ where each I_k constrains how I_{k-1} can evolve. These may be sets of value functions, acceptance regions in policy space, or formal contracts about subsystem behavior. Self-modification is represented by a typed self-map T on a state space that includes mechanisms, invariants, and any environment variables needed to make the map closed.

Banach case. Let (X, d) be a nonempty complete metric space, let $T : X \to X$ be a self-map, and suppose there is a constant $q \in [0, 1)$ such that

$$d(Tx, Ty) \leq q\, d(x, y) \qquad \text{for all } x, y \in X. \tag{49}$$

Then Banach's fixed-point theorem gives a unique fixed point x^* and convergence of the Picard iteration $x_{n+1} = T(x_n)$ from every $x_0 \in X$[26]. Applying this result to self-modification requires proving completeness, self-mapping, and the contraction bound for the actual state and metric; intentional adherence to a playbook is not enough by itself. The theorem establishes mathematical convergence in d, not alignment or safety unless those properties are separately encoded and proved to follow from the metric and invariant region.

Schauder case. Let C be a nonempty compact convex subset of a locally convex topological vector space, and let $T : C \to C$ be continuous. Schauder's theorem gives at least one fixed point[27]. It does not by itself provide uniqueness, convergence of an iteration, Lyapunov stability, a computable fixed point, or a safety result. For open-ended modification, a Schauder model is therefore an existence hypothesis plus an engineering monitoring programme, not a convergence certificate.

Claim	Required evidence	What follows
Banach fixed point	Complete metric space, self-map, global or stated local contraction $q < 1$	Unique fixed point and Picard convergence in the declared region; no independent safety conclusion.
Schauder fixed point	Nonempty compact convex domain in a locally convex space and continuous self-map	Existence of at least one fixed point; no uniqueness or iteration convergence.
Stable attractor	Dynamics, invariant neighborhood, and a proved or empirically calibrated stability condition	Only the stated local or probabilistic stability conclusion.
Governed safety	Threat model, protected properties, enforcement, coverage, residual risk, and external validation	A scoped assurance statement, never a consequence of a fixed-point theorem alone.

Table 9: Fixed-point and stability claims are tied to their actual hypotheses and conclusions.

11.3 Global Regulators: Consistent Principles Across All Cognition

Throughout the new incarnation of PRIMUS, two mathematical principles guide every decision:

Geodesic effort supplies a common selection score for inference, learning, planning, and self-modification, balancing forward reachability with backward usefulness per declared unit cost. It biases search toward economical, goal-relevant steps and makes the cost model inspectable. It does not by itself establish a global minimum, eliminate waste, or prove faster search; those outcomes are benchmarked for each instantiated process.

Weakness-based simplicity provides a universal preference for simpler solutions. Whether choosing between proof strategies, neural architectures, or system modifications, the weakness measure quantifies "failure to distinguish" and creates consistent pressure toward more general, robust solutions.

When these principles are used to favor moderated and beneficial goal evolution, they can influence many algorithms and structures in the system. Seeking simple representations of system goals and using the canonical geodesic score to select bounded-cost steps toward futures in which those goals are substantially realized gives PRIMUS more explicit handles for maintaining ethical behavior as it self-modifies. The effect remains contingent on the motive representation, metrics, policy, threat model, and empirical performance of the controls.

The objective is an approach in which increased capability need not imply decreased predictability: the system can become more powerful while retaining measurable, reviewable, and enforceable control surfaces.

11.4 The Self-Modification Pipeline

11.4.1 Beyond Hot-Patching

Self-improvement is not treated as a sequence of unreviewed quick fixes. The proposed five-stage pipeline treats each modification as a governed experiment:

1. **Proposal** - Generate a candidate change (new algorithm, data structure, or module).
2. **Analysis** - Examine theoretical impact through type checking and dependency analysis.
3. **Simulation** - Test in a digital twin environment with reduced but representative workloads.
4. **Scoped validation and admission** - Produce a report stating which invariants and performance criteria passed, under which assumptions, coverage, and margins.
5. **Deployment** - Stage rollout with monitoring, capability restrictions, and scoped recovery options.

11.4.2 Technical Details

Each modification is formalized as a typed endomorphism, or more generally a typed system transformation, $\theta : C \to C'$ between declared component-state spaces. The term "metamorphism" is not used as a substitute for a standard typed map. The signature includes:

- Pre- and post-conditions that must be satisfied
- Promised improvements in complexity or performance
- Expected changes in weakness (simplicity) metrics

Static analysis builds an influence graph showing which components are affected. The system checks applicable lens laws and typed composition conditions, and records whether the modification preserves each declared structural property within the analysis scope.

Twin simulation creates a controlled copy of relevant system components. Using Weighted Atom Sweeps to select representative regions, the system replays declared workloads while measuring:

- value-function divergence from original behavior;
- weakness trajectory and the definition under which it is compared;
- order defects and commutativity penalties for named operators; and
- evidence-accounting integrity, replayability, omissions, retractions, and duplicate handling.

A twin cannot prove that its workload is representative. The report therefore states coverage, sampling, excluded side effects, simulator fidelity, and known distribution gaps.

Scoped validation and admission evaluates the declared criteria:

- invariant levels I_k remain within prescribed bands on the tested distribution;
- behavioral drift stays below a named threshold: $\texttt{div}(\texttt{old}, \texttt{new}) \leq \varepsilon$;

- weakness and other objectives meet their declared acceptance rules; and
- evidence records remain ordered, replayable, non-duplicative, and linked under the logging policy.

The output is a scoped validation report documenting assumptions, versions, coverage, confidence, margins, residual errors, what transferred, what must be relearned, expiration, and monitoring. It is not a general safety certificate.

Staged deployment uses RSpace/Rholang executors with capability-based security. Changes initially run in shadow mode alongside the original, graduate to dual-run with comparison, then elevate to primary when named metrics and review criteria pass. Patches, reports, and metrics are content-addressed for integrity and linkage; signatures, authorization, availability, and semantic validation remain separate properties.

11.5 OmegaSelf Continuity Gating: Making the Pipeline Enforceable

The five-stage self-modification pipeline becomes operationally meaningful only when it is connected to an execution choke point. OmegaSelf provides that connection. A self-modification is first normalized into an immutable typed proposal whose canonical hash covers the full intended change, affected namespaces and capabilities, policy version, predicted effects, rollback plan, and expiry. A vague approval for a target name or a previous version of the proposal cannot authorize a different body.

For a high-impact change, the policy consumer should require:

1. a verified current `SelfHereNow` anchor appropriate to the risk tier;
2. a pre-change snapshot, evidence-ledger root, and structural signature;
3. a complete evidence closure and committed prediction of the post-change behavior;
4. explicit protected invariants, commitments, and authorization sources;
5. twin or counterfactual test evidence that remains quarantined from live claims;
6. an exact-input continuity validation report, or an independently reviewed conservative report scoped to the mutation class, with assumptions, coverage, residual error, and expiry;
7. an approved exact proposal with a short validity interval;
8. execution and post-change receipts, a transport record, and a tested rollback path.

Continuity analysis and governance remain separate. The continuity service may report that a proposal is verified, contradicted, incomplete, unsupported, budget-exhausted, or absent from cache. Only policy maps that result to `Allow`, review, deferral, or denial. A cache miss, heuristic extension, or failed analysis budget cannot authorize a protected mutation. This matters because the deepest continuity checks may be expensive; latency pressure must not turn incomplete analysis into tacit permission.

The resulting transport does not claim that the post-change system is strictly identical to the pre-change system. It records ancestry, preserved witnesses, failed and preserved invariants, reconciliation policy, information loss, authorization, reversibility, and rollback reference. If rollback is invoked, OmegaSelf creates another directed transport from the failed branch to a restored state. The intervening episode remains in the causal and evidential history, which is essential for learning from the failure and for preventing an apparent restoration from hiding later commitments or actions.

This operational layer also resolves a subtle governance problem. The system may use PLN, weakness, geodesic control, TransWeave, and invariant analysis to estimate whether a change is

desirable. None of these estimates, alone or in combination, should execute the change. They produce evidence and proposals. Authority comes from the externally rooted, versioned policy context and the exact proposal decision. Thus self-reflection can become increasingly sophisticated without allowing the reflective model to manufacture its own permission.

11.6 Safety Through Transparency

11.6.1 Making Safety Checkable

Every component and modification carries explicit assurance documentation that higher-level processes can inspect:

Type interfaces specify behavioral contracts including domains, codomains, preconditions, post-conditions, effect bounds, and monotonicity properties where relevant.

Scoped admission reports include:

- SubRep reports stating the motive slice, distribution, coverage, margin, and residual risk for a new option; and
- TransWeave reports stating typed maps, order and value residuals, transfer coverage, and regions requiring relearning.

Neither report establishes indefinite composability or whole-system safety.

Drift budgets establish tolerance bands for each invariant level. A supermartingale claim is made only after defining a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t), \mathbb{P})$ and an integrable adapted potential Φ_t such that, in the declared validation regime,

$$\mathbb{E}[\Phi_{t+1} \mid \mathcal{F}_t] \leq \Phi_t - \kappa_t, \qquad \kappa_t \geq 0. \tag{50}$$

A Lyapunov claim similarly names the dynamics, state domain, potential, and decrease condition. Operational dashboards estimate these quantities with uncertainty; they do not convert a finite test into a theorem. If the estimate rises or confidence becomes inadequate, policy may reduce step size, hold deployment, require review, or invoke scoped recovery.

11.7 Decentralized Governance

11.7.1 Multi-Party Evolution

Real-world deployment often involves multiple teams, organizations, or even multiple AI systems collaborating on improvements. Hyperon's architecture supports this through:

Capability security - Self-modification jobs compile to Rholang processes running under RSpace with object-capability guards. Edit permissions are precisely scoped to specific Spaces, prefix ranges, or component classes.

Cryptographic commitment, identity, and provenance records - Every modification artifact can receive a CID. A CID binds canonical content bytes under a stated multicodec, multihash, and encoding; it does not by itself prove author identity, causal provenance, factual truth, authorization, or continued availability[31]. Signatures bind a key to bytes under key-control and identity assumptions. FireNode/F1R3FLY can post signed summaries to chain, establishing that a commitment was recorded and ordered under the chain's consensus assumptions. Execution receipts, semantic validators, authorization policy, availability services, and dispute/correction procedures supply the remaining properties.

Scoped recovery - ShardZipper's Merkle-based structure and SMS macro-patches can restore named checkpointed internal components to a selected known-good state. A rollback trigger is not universal or necessarily instantaneous. It cannot erase external communications, disclosures, on-chain actions, financial settlement, physical effects, or downstream copies, and it does not undo commitments made between the checkpoint and recovery. Each deployment therefore states the checkpointed components, consistency boundary, recovery-point objective (RPO), recovery-time objective (RTO), retained side-effect log, data-loss expectation, dependency order, and compensating actions for irreversible effects. Recovery is tested by drills and measured distributions rather than promised by the existence of a checkpoint.

11.8 Illustrative Target Scenario: Upgrading Predictive Coding

11.8.1 Walking Through the Proposed Pipeline

The following numbers are illustrative targets and acceptance criteria, not observed performance results. The scenario shows how a reproducible trial would be specified for a predictive-coding component that uses WILLIAM-guided sparsity.

Claim status: Illustrative target scenario; no success result is claimed without timestamped code, data, hardware, raw measurements, and signed evaluation artifacts.

Proposal target: Replace dense attention with top-k selection guided by compression metrics. The candidate objective is 30% fewer FLOPs while limiting task-metric loss to no more than 2% on a predeclared benchmark suite. The baseline, metric, confidence interval, and hardware are part of the trial specification.

Analysis target: Construct the influence graph for pattern-head calls, PLN hooks, and mining iterators, and test the declared ByteFlow patch and ShardZipper lens properties. Passing a lens test covers the specified transformations; it does not validate unrelated behavior.

Simulation target: Clone hot prefixes identified by WAS and allocate at most 25% of the trial's declared geodesic compute budget. Monitoring reports:

- value drift under a named bisimulation or task metric;
- weakness trajectory under the instantiated weakness definition;
- order-defect distributions for the named neural update operators; and
- calibration, failures, tail latency, and simulator-to-live fidelity.

Admission target: A successful trial would show that tested options in domains A and B remain within their SubRep report margins; that two specified H-ICA components meet target-domain transfer bounds while one is correctly rejected for relearning; that drift remains below a predeclared ε; and that any supermartingale statement satisfies Equation 50 with reported estimation uncertainty.

Deployment target: Run in shadow mode for a planned 24-hour observation window, then in dual-run mode, and elevate only after predeclared KPIs and human or policy review pass. The last-good CID identifies the checkpoint bytes. Recovery time, data loss, checkpoint scope, irreversible side effects, and compensating actions are measured under the RPO/RTO plan; the CID does not make rollback instantaneous.

A successful, reproducible outcome would support the claim that the modified component is faster or simpler within the tested scope while preserving the named downstream metrics. It would not establish that all safety properties or all proof and planning behavior are preserved in every environment.

11.9 Implementation Checklist

11.9.1 Building Self-Improvement Capabilities

To implement safe self-modification:

Goal representation: Build a library for invariant hierarchies with support for multiple metric families (bisimulation, f-divergences, weakness). Expose `check_invariant(k, bands, sample)` as a grounded operation.

Geodesic control: Create a shared service that meters effort across cognitive operations using Equation 30. Provide a typed API such as `score_step(state, action, baseline, forward, backward, cost)` that validates positivity, normalization, baseline identity, and $c_x(a) > 0$, and returns $S_x(a)$ with uncertainty metadata.

Weakness tracking: Store $Q_{\texttt{logic}} \times Q_{\texttt{tv}}$ valuations with all Atoms. Make per-edge weakness deltas available to every scheduler. Use weakness as the universal regularizer.

Modification pipeline: Implement:

- Typed edit proposals with lens checking
- Influence graph construction
- Twin environment creation and management
- Divergence and commutativity meters
- Scoped validation-report generation
- Staged rollout orchestration

Decentralized execution: Create Rholang templates for capability-controlled modifications, FireNode hooks for posting signed validation reports, and SMS integration for reproducible rollback.

Monitoring dashboards: Build live displays for supermartingale potentials, divergence bands, weakness trends, and geodesic budget consumption–with one-click hold/elevate/recovery triggers, with authorization and displayed RPO/RTO scope.

11.10 The Core Insight

11.10.1 Unity of Principles

The profound insight underlying this approach is that the same mathematics that governs routine cognition can also govern self-modification. If daily reasoning follows geodesic paths, so should system upgrades. If simplicity (weakness) helps generalization in learning, it should also guide architectural evolution. If goal decomposition through MetaMo/SubRep works for planning, it should also work for validating new capabilities.

This is practical engineering rather than philosophical symmetry alone. Using the same frameworks throughout creates a disciplined way to test whether a proposed improvement raises capability without exceeding declared alignment, interpretability, authority, and human-value constraints. The common framework does not guarantee those properties; it makes them explicit enough to measure, challenge, and enforce within a stated scope.

The path from AGI toward ASI is therefore designed as a sequence of typed, reviewed, staged, and–for named checkpointed internal components–recoverable steps. Each step must demonstrate which properties it preserves, which residual risks remain, and which external effects cannot be reversed.

Mathematics supplies explicit hypotheses and conditional results; engineering supplies control surfaces; accumulated evidence determines when additional trust is warranted.

12 Example Application Domains: From Current Pilots to General Intelligence

Our development philosophy is straightforward: build real applications in diverse domains from day one, using each as both a testbed and a source of learning that benefits the others. The current programme includes domain pilots in game AI, humanoid social robotics, bioinformatics hypothesis generation, and mathematical conjecturing with formal proof, plus OmegaHive as a cross-domain operational laboratory for research, software, formalization, writing, governance, and collective cognition.

The four domain pilots exercise different combinations of perception, reasoning, prediction, planning, motivation, and action. OmegaHive exercises the control fabric itself: task decomposition, module and agent specialization, shared memory, provenance, permissions, human escalation, and architectural learning. All of them connect to the same Hyperon and PRIMUS substrate. Advances in neural-symbolic bridging, PLN, predictive coding, WILLIAM, attention, Context Frames, and self-modification can therefore be reused across substantially different workloads rather than rebuilt in separate technology stacks.

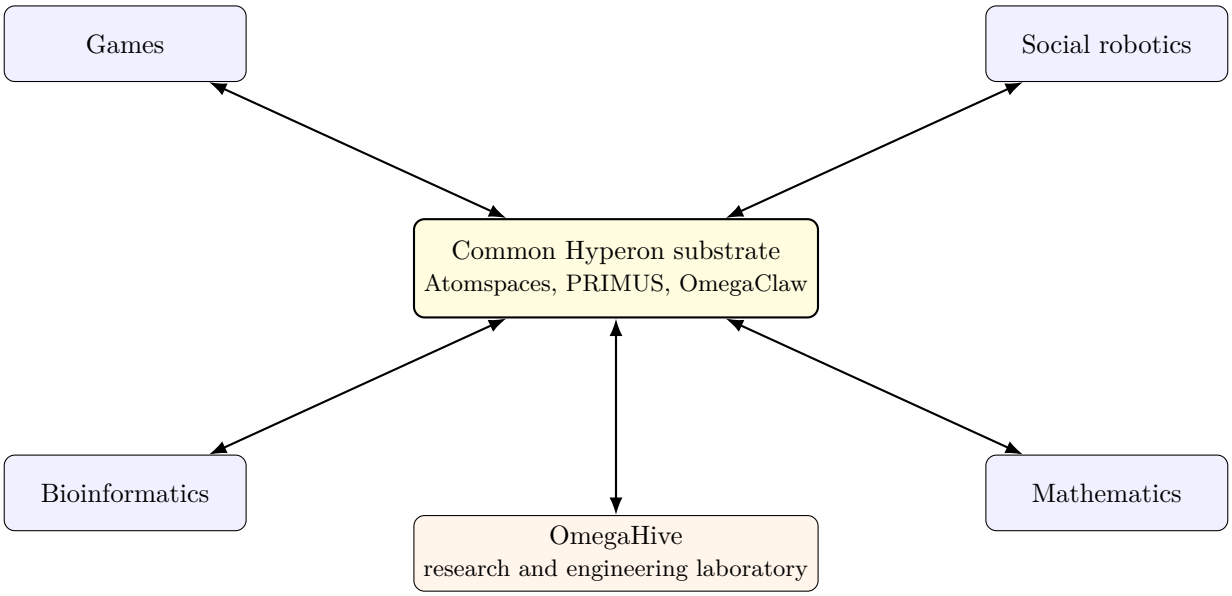

Figure 10: Domain applications and OmegaHive share a common Hyperon substrate while retaining local task state and specialized modules.

12.1 OmegaHive: Research and Engineering as a Cognitive Workload

OmegaHive is both infrastructure and an application. Its practical workload–literature research, mathematical formalization, software development, technical writing, editing, curation, and system operations–is unusually well suited for evaluating a general cognitive control fabric because the outputs are concrete and heterogeneous. Papers can be checked for sources, proofs can be checked by Lean, code can be tested, task histories can be audited, and human operators can judge whether the collective is producing insight or merely generating activity.

A representative workflow begins with a research goal entered into a shared Context Frame. A chief-of-staff OmegaClaw decomposes the goal into literature search, conceptual analysis, formalization, implementation, experiment, and writing tasks. Specialist agents claim tasks on the board. A librarian curates source artifacts. A mathematical agent submits candidate theorems to a persistent proof service. A coding agent implements experiments. Writer agents draw from raw task and bus records, and an editor gates completion on a valid notes-and-sources companion. A conscience process monitors value and organizational dynamics, while deterministic scripts track token spend, blocked work, latency, rework, and escalation.

The same workflow can progressively become more PRIMUS-native. PLN can assess claims and evidence. WILLIAM can mine recurring research or proof patterns. SubRep can learn task decompositions admitted under scoped reports. ECAN can allocate attention across the board. MetaMo can represent tradeoffs between novelty, rigor, timeliness, and cost. The predictive-coding transformer bridge can support local research and writing models whose symbolic heads retrieve the shared ontology and provenance records. Useful coordination patterns can be promoted from prompts and role descriptions into MeTTa rules and Module Spaces.

OmegaHive therefore provides a measurable bridge between today's tool-using agents and tomorrow's distributed Hyperon cognition. Success is not defined by the number of agents or messages. It is defined by higher-quality artifacts per unit cost, better calibration and provenance, lower rework, reliable permission boundaries, effective human escalation, and architectural lessons that transfer back into OmegaClaw and PRIMUS.

12.2 Game AI: From Minecraft to Sophiaverse and the Neoterics Playground

12.2.1 The Opportunity

Games provide controlled sandboxes for testing perception, planning, tool use, dialogue, and social norms with rapid iteration and comparatively low-cost failure. Minecraft offers a structured yet open world well suited to integrating SubRep options with GEO-EVO edits. Sophiaverse adds persistent identity, social contracts, and economic systems. Within Sophiaverse, the project is designing a specialized "Neoterics" micro-world: a deliberately constrained but richly instrumented environment for early-stage AGI development, with short feedback loops, dense sensor coverage, and low-cost resets for rapid training and testing of PRIMUS components.

12.2.2 Technical Implementation

OmegaClaw and OmegaHive agents can serve as effective wrappers for NPCs in games, allowing AI agents to act with sophisticated self, memory, reasoning and social interaction in diverse game contexts.

Behind the scenes, the world interface operates through Spaces, mirroring game state (voxels, entities, quests, markets) into Atomspace as typed Atoms. Action affordances appear as SubRep options (navigate, craft, trade, negotiate) with MeTTa rules expressing pre/post-conditions. Each option admission receives a scoped CDS/PDS report stating assumptions, coverage, margins, version, and residual error, and the associated bytes are stored with CIDs for integrity linkage and reproducible retrieval. Semantic validity and continued availability remain separately checked.

For perception and policy, we initially use external LLMs for narration and quest generation, with Symbolic Heads retrieving mined templates like task frames, recipes, and negotiation scripts. Inside the graph, QuantiMORK hosts lightweight PC waveformers that locally refine spatial or dialogue

embeddings. WILLIAM steers attention by managing top-k heads, tokens, and scales while prioritizing unification order based on heavy-edge importance.

Reasoning operates through PLN factor graphs that encode precondition networks and multi-step plans. Geodesic control scores candidate steps with $S_x(a)$ from Equation 30, using the same normalization, baseline, and cost conventions for MOSES edits in programmatic policies.

Learning and transfer work through GEO-EVO, which couples MOSES demes (for crafting plans and trading tactics) with backward goal factors. TransWeave proposes successful options and value maps for transfer from Minecraft to Sophiaverse/Neoterics when typed H-ICA diagnostics, target-domain tests, and residual budgets support the mapping. Weakness gates are intended to contain brittle reuse; avoided-failure and false-admission rates measure whether they do.

The runtime leverages Weighted Atom Sweeps to maintain salience, ByteFlow PPTNs to pack hot subtries for GPU kernels, and ShardZipper to batch world-tick updates deterministically. Consider a micro-flow in Neoterics for "build-and-barter" gameplay: The pattern miner discovers the motif ⟨mine→smelt→craft→trade⟩. PLN assembles a three-step plan. Geodesic control selects the cheapest next step. The PC waveformer refines local perception only where surprise exceeds threshold. WILLIAM prunes attention heads and tokens for efficiency. Upon successful completion, TransWeave logs an option map that Sophiaverse agents can reuse with adjusted prices.

Near-term milestones include a Symbolic Head MVP for quest/recipe frames with weakness-aware pruning, PLN + SubRep integration for 5-10 canonical Minecraft tasks with GEO-EVO corridor demonstrations, and a Neoterics "micro-agency" loop achieving sub-second decisions with PC local refinement and WAS scheduling.

12.2.3 Action-First World-Model Extension

The game agent should maintain more than a mirrored voxel database. Its world model is an action-indexed affordance landscape: what can be mined, combined, traded, activated, avoided, or tested, under which environmental and social conditions, with what predicted outcome. The game API and rendered observations enter S_{env} and S_{evid}; persistent entities and map hypotheses live in S_{ent} and S_{map}; uncertain recipes, norms, and affordances live in S_{rule}; and HMH records index structured experiment episodes, skills, and comparable situations.

Unknown affordance discovery is a key benchmark. When a modded item has an uncertain role, the agent stores competing hypotheses such as conductor, trigger, insulator, or context-sensitive catalyst. It retrieves similar trials through HMH, uses PLN abduction to identify missing preconditions, and chooses controlled experiments that discriminate among the hypotheses. The fast path handles navigation and hazards; the mid-level loop updates context and selects trials; the ambient loop consolidates successful experiments into rules, programs, HMH templates, and SubRep options.

Staleness matters because game worlds change. Map facts, entity states, recipes, and affordance models receive time- and hazard-dependent applicability. A low-confidence or stale hypothesis creates a revalidation option instead of being silently forgotten. OmegaClaw's Context Frame records the current action domain, evidence, retrieved cases, competing worlds, experiment plan, predictions, and results. Procedural world-generators compiled from PLN proofs can sample alternative causal branches before an expensive or irreversible action, while keeping those worlds in counterfactual quarantine.

This world-model benchmark is more informative than task completion alone. Evaluation should measure identity tracking, action-prediction calibration, sample efficiency of affordance discovery, staleness recovery, option transfer, counterfactual discrimination, and whether the agent can explain which evidence changed its model.

12.3 Humanoid Social Robotics: Education and Performance

12.3.1 The Partners and Vision

Current collaborations include two robotics companies. Mind Children is developing a 3.5-foot education robot for classroom interaction. Hanson Robotics, creator of Sophia and Desdemona, contributes experience with expressive humanoid robots for public engagement. These partnerships provide demanding integration tests for PRIMUS: a robot must combine perception, dialogue, motor control, and social norms within one deployment. The cognitive substrate must support tasks ranging from explaining fractions to a third-grader, to comforting an anxious student, to performing on stage without requiring a separate architecture for each scenario.

12.3.2 Technical Architecture

Perception begins with audio and vision neural networks running externally, while PC layers can progressively be embedded through QuantiMORK for latency-critical paths such as gaze estimation, lip synchronization, and gesture segmentation. Prediction errors feed PLN hypotheses such as "student is pointing at X" and "person is asking a question," while symbolic constraints steer refinements and expand only the frequency bands tied to current hypotheses. For dialogue and social scripts, Pattern Miner discovers conversational templates including adjacency pairs and tutoring frames. Symbolic Heads allow the speech model to retrieve these patterns during decoding, while PLN enforces persona and safety constraints. MetaMo budgets the balance between exploration and de-escalation behaviors, throttling novelty when emotional risk is high.

Motor control keeps low-level servo loops on the robot while SubRep options cover higher-level skills such as "look at face," "nod," and "hand wave." Geodesic control aligns dialogue steps with motion options by ranking them with $S_x(a)$ from Equation 30, so physical gestures support conversational goals under a declared cost model. Weakness penalties are intended to discourage overcommitted or awkward gestures.

Safety and alignment mechanisms use Meta-Goal Stability to track drift in social norms and persona. Commutativity penalties measure whether PC "sensation" updates and RL "decision" updates conflict. RSpace executors and lens contracts gate what an on-device agent may modify. The runtime uses WAS for background maintenance (belief cleanup and memory decay), ShardZipper to extract "interaction shards" (short temporal windows) for batch prosody and gesture retiming, and ByteFlow to handle drifting hot regions in live scenes.

A typical classroom tutoring micro-flow is illustrative. A student asks a question. ASR converts it to PC features, generating PLN hypotheses about intent. A Symbolic Head retrieves the appropriate tutoring frame. MetaMo selects an "explain plus gesture" option bundle. A geodesic-ranked step executes "point to board, then speak." PC refines lip and gaze locally for natural interaction. WILLIAM prunes unused heads for efficiency. Evidence capsules record source identity and replay links to reduce duplicate counting.

Near-term milestones include inside-graph PC for gaze/lip/speech synchronization with commutativity regularization, a SubRep library of 20-30 gesture/dialogue options with scoped admission reports, and a safety dashboard showing invariant bands for persona/etiquette, drift budgets, and rollback controls.

12.3.3 OmegaClaw Memory and PRIMUS World Modeling in Mind Children

OmegaClaw is now being used in the Mind Children robots to provide long-term memory across interactions and sessions, and this role is expected to expand. The immediate benefit is continuity: a classroom robot can remember people, names, prior explanations, preferences, commitments, unresolved questions, and the outcomes of earlier tutoring episodes rather than treating each conversation as a fresh prompt. The larger architectural significance is that this memory becomes the persistent evidence and episode layer of a broader PRIMUS world model.

The robot's world model is organized as the multi-rate braid in Section 5. Audio, video, depth, pose, tactile, and proprioceptive traces are selectively stored as content-addressed evidence. Persistent person, object, place, and event hypotheses retain links to that evidence. HMH encodings index structured interaction episodes, tutoring frames, gesture contexts, social scripts, and skill invocations, preserving roles such as teacher, student, speaker, listener, requested topic, emotional valence, norm, and commitment. Dense models handle perception, dialogue surface form, and motor dynamics, while PLN represents explicit norms, causal hypotheses, schedules, permissions, and uncertainty.

Action-first semantics provide the quality criterion. The robot's current state is not merely a list of recognized objects and remembered facts. It is a distribution over available external and cognitive actions: look again, ask a clarifying question, retrieve a prior explanation, point to an object, change speaking style, pause, seek a human, or execute a gesture admitted under a current scoped report. For each action family, OmegaClaw can maintain predictions of success, social effect, latency, energy, risk, and expected frame change. A remembered episode is valuable when it improves these action-conditioned predictions, not merely because it is semantically similar to the current conversation.

A Context Frame binds the task-relevant world-model slice for one interaction. It includes the identified participants, evidence and identity confidence, dialogue and task goals, retrieved HMH episodes and scripts, explicit commitments, candidate interpretations, current affordances, predicted outcomes, safety constraints, and pending actions. Module Spaces expose ASR, vision, speaker and face identification, dialogue models, predictive-coding controllers, PLN, long-term memory, and motor skills behind typed interfaces. The Frame remains authoritative across model calls and can be inspected by teachers, developers, and safety processes.

The world model runs at three rates. Low-level balance, collision avoidance, gaze stabilization, lip synchronization, and servo control remain in the fast S_{dyn} path. Mid-level cognition retrieves relevant episodes, updates identity and intent hypotheses, checks norms, selects SubRep options, and coordinates speech with gesture. The ambient loop consolidates repeated tutoring patterns, revises person models, mines social and pedagogical motifs, updates HMH templates, and schedules revalidation of stale beliefs. If the Atomspace or network is busy, hard real-time safety remains local.

Evidence anchoring is particularly important in human interaction. A fact such as "the student is anxious" should be an uncertain, context-specific hypothesis linked to voice, facial, posture, dialogue, and situational evidence, not an unqualified personality label. Later observations may support, weaken, or reinterpret it. ωPLN can retain positive and negative evidence in separate contexts, avoiding the collapse of conflicting classroom, family, and stage behavior into one global persona estimate.

Procedural world-generators provide structured counterfactuals. From a PLN proof that a student may be confused, the robot can sample witness-worlds in which the cause is missing prerequisite knowledge, ambiguous wording, distraction, hearing difficulty, or social anxiety. Each world predicts different observations and suggests a different cognitive action. The system can choose a low-risk clarifying question or demonstration with high expected information value. Generated worlds remain hypothetical and do not count as new evidence.

TECAN can allocate typed resources among dialogue, perception, memory retrieval, abduction, causal replay, gesture planning, and human review. A socially sensitive abduction should not draw from the same unrestricted pool as cheap text completion. Semantic chemistry can mine and test recurrent interaction loops–for example, clarification, explanation, gesture, feedback, and repair– while causal ablation distinguishes genuinely helpful tutoring scripts from loops that merely keep the conversation active.

A representative extended tutoring flow is:

1. A student asks a question; ASR and vision append evidence and renew the participant identity hypothesis.
2. OmegaClaw retrieves relevant prior interactions, HMH tutoring episodes, and explicit commitments into the Context Frame.
3. PLN and neural classifiers maintain several intent and confusion hypotheses, each with provenance and predicted observations.
4. A proof-derived world generator produces a small set of causal witness-worlds and candidate clarifying actions.
5. MetaMo and policy select an explain–demonstrate–gesture bundle; SubRep supplies high-level actions admitted under current scoped reports, while the local motor stack enforces its separately tested hard-safety constraints.
6. Predictive coding refines gaze, prosody, lip motion, and gesture timing. The execution receipt and student response close the prediction loop.
7. The episode is stored with evidence, action, outcome, and discrepancy. Ambient cognition later consolidates it into person-specific memory and more general tutoring patterns.

Near-term milestones include robust multi-session memory with provenance and privacy scopes; person-specific retrieval that distinguishes observed facts from inferred traits; a world-model Context Frame shared by dialogue, perception, and motor planning; HMH indexing of episodes, scripts, and options; calibrated action-prediction and revalidation; and dashboards showing identity confidence, memory sources, social and persona constraints, open commitments, staleness, permission, prediction error, and rollback state.

12.4 Bioinformatics: Graph-Based Discovery for Longevity and Beyond

12.4.1 Why Hyperon Excels Here

Biology is fundamentally graph-structured: genes connect to proteins, proteins form pathways, pathways influence phenotypes, and drugs modulate these relationships. Longevity research benefits not from indiscriminate processing of every dataset but from forming testable hypotheses about targets, combinations, and mechanisms. Hyperon is well suited to combining noisy graphs, mining candidate motifs, running uncertain chains of reasoning, and proposing ranked, testable hypotheses with explicit rationales for prioritization.

12.4.2 System Architecture

Data flows into Atomspace through BioSpace adapters that transform omics matrices into node attributes, protein-protein interactions and pathways into edges, literature triples into assertions with

provenance, and clinical outcomes into noisy links with confidence scores. Everything receives CIDs, making merges auditable and reproducible.

Pattern discovery operates through Pattern Miner, which identifies motifs like "gene A $\leftrightarrow$ pathway P $\leftrightarrow$ phenotype Y with drug D evidence", ranking them by I-surprisingness and support. WILLIAM transforms frequent subgraphs into reusable templates representing candidate mechanisms.

Reasoning and causality flow through PLN factor graphs that propagate graded truth over ontologies, experimental results, and pathway priors. AIRIS-style causal-induction constraints filter implausible relations. A geodesic planner assembles mechanism candidates using the canonical $S_x(a)$ score, balancing reachability from data with usefulness for therapeutic objectives under a declared cost band.

For programs and predictions, MOSES/GEO-EVO evolves small predictive programs (risk scores, treatment effect surrogates) guided by SubRep subgoals like "improve precision on aged cohort". Weakness (Occam's razor) controls model complexity, with models stored as MORK objects including evaluation capsules.

Transfer operates through TransWeave, which proposes mechanism components across cohorts or omics platforms when typed H-ICA diagnostics, target-domain tests, and residual budgets support the mapping. Impossibility and fail-fast flags force partial transfer or fresh learning when appropriate, with brittle-reuse reduction measured rather than assumed.

The runtime uses ShardZipper to extract cohorts or pathway-centric shards, ByteFlow to accelerate repeated motif scans, and WAS to schedule refresh where new data arrives.

Consider an example combination-therapy hypothesis micro-flow. The miner highlights a motif linking mTOR and autophagy through two drug actions. PLN infers a plausible double-hit mechanism under aging markers. MOSES proposes a response predictor. The geodesic planner selects the next assay step by maximizing $S_x(a)$ from Equation 30. TransWeave proposes matched components from related data sets subject to target-domain validation. The output is a ranked hypothesis pack: a CID-linked bundle containing the mechanism graph, predictor, expected biomarkers, and counter-evidence.

Near-term milestones include an end-to-end hypothesis generator on a large longevity dataset combining omics and literature, GEO-EVO corridor using the canonical $S_x(a)$ guidance for experiment selection and scoped SubRep admission reports for assay steps, and a weakness-aware model portfolio emphasizing few, simple, transferable models with full audit trails.

12.4.3 Semantic Chemistry and Procedural Hypothesis Worlds

Semantic algorithmic chemistry provides a natural ambient discovery layer for biological graphs. Pathway motifs, phenotypes, intervention schemas, and evidence contexts become semantic molecules; contextual PLN implications become reaction rules; and recurrent multi-mechanism loops become candidate ACSs. Causal replay and ablation test whether a loop actually changes hypothesis ranking or predicted outcomes rather than merely reinforcing correlated literature. TECAN budgets mechanism generation, trial design, provenance checking, and causal replay separately.

PLN proof-to-generator compilation turns a ranked mechanism proof into explicit hypothetical patient or experimental worlds. Intermediate biological Atoms become latent mechanisms; deduction and abduction become kernels; distributional truth values can produce heterogeneous response worlds; and predicted observables become candidate biomarkers or assay outcomes. The worlds remain provenance-linked simulations, not clinical evidence. They are useful because they make the hidden alternatives behind one scalar recommendation visible and suggest discriminating experiments.

12.5 Mathematical Theorem Proving with Automated Conjecturing

12.5.1 Beyond Proof to Discovery

Most automated theorem proving focuses on stated goals, while mathematical work also depends on conjecturing–formulating definitions, lemmas, and "what if" hypotheses that are later proved, refuted, or refined. The programme therefore targets automated conjecturing as a primary objective and uses formal proof for validation. A formal proof verifier has recently been implemented in MM2 inside MORK, providing a graph-local proof kernel; performance is to be reported through versioned benchmarks rather than characterized by an unscoped speed adjective.

12.5.2 Technical Design

OmegaClaw-Math treats the persistent proof project, rather than an LLM conversation or individual model instance, as the enduring mathematical agent. Each project is represented in a MORK-backed PeTTa metagraph, supported by a revisioned append-only event journal and a content-addressed artifact store. The metagraph preserves theorem statements, definitions, informal claims, exact formal declarations, alternative proof strategies, failed attempts, experiments, formal proof states, verified lemmas, counterexamples, critic reports, provenance, and human guidance under stable identifiers. Large objects such as Lean source files, proof traces, model outputs, computational notebooks, and diagnostic logs remain in the artifact store and are referenced by content hash.

The architecture organizes mathematical work into two nested loops operating at different timescales:

- The outer research loop is coordinated by OmegaClaw-Math. It selects which claim, route, obstruction, experiment, or formalization task deserves attention; compiles a role-specific context packet; dispatches bounded LLM, Hyperon, critic, analogy, experiment, or statement-alignment workers; and interprets their results relative to the literal theorem.
- The inner formal loop is a Lean-native automated theorem-proving service. It receives an exact Lean declaration, a pinned software and library environment, an accessible premise set, a model bundle, and a finite search budget. GNN policies predict tactics, arguments, and relevant premises; PLN and related mechanisms provide uncertain relevance and search annotations; Pantograph applies proposed tactics to exact Lean states; and a local AND/OR proof hypergraph organizes the resulting search.

The two loops share one typed proof metagraph but retain different state machines and authority boundaries. The outer loop works with research moves such as "invent a bridge lemma," "replace the induction invariant," "test a possible counterexample," or "change the representation." The inner loop works with exact tactic applications under specific Lean states. A single formal run may contain thousands of local tactic transitions without requiring OmegaClaw to supervise each one. Conversely, the formal ATP service cannot silently change the theorem, weaken assumptions, introduce a new informal mechanism, or decide that a nearby result is good enough.

The unified MORK metagraph contains several linked subgraphs:

- a research search graph of informal and semiformal states, moves, routes, alternatives, barriers, generalizations, bypasses, and reformulations;
- a formal proof-search graph of exact Lean states and tactic-application hyperedges;

- a statement-alignment layer connecting informal mathematical claims to exact Lean declarations while recording quantifiers, definitions, domains, assumptions, constructivity conditions, and any strengthening or weakening;
- a justification graph linking claims to dependencies, critics, experiments, counterexamples, and formal certificates;
- a provenance graph recording attempts, contexts, prompts, models, corpora, software environments, tactic traces, artifacts, and replay results;
- a speculative layer containing unverified Hyperon and LLM hypotheses; and
- derived views for active frontiers, dependency closures, reusable verified lemmas, repeated obstruction families, transpositions, route statistics, and taint propagation.

The formal ATP service may maintain a fast run-local Python hypergraph, but this cache is not authoritative. It periodically emits durable checkpoints containing exact state hashes, tactic edges, complete child lists, frontier summaries, model and environment identities, trace artifacts, and resource usage. MORK is the project-level semantic memory, the event journal is the replay and durability authority, and the artifact store is the byte-level authority for large objects.

Formal proof search uses strict AND/OR semantics. An exact Lean state is an OR node: any one valid outgoing tactic may solve it. A tactic application is an AND edge: every subgoal returned by Lean is a required child. If Lean returns five goals, all five are stored even when the scheduler initially expands only one. A tactic accepted with zero remaining goals creates a genuinely solved leaf. A rejected tactic, timeout, parser failure, missing backend, or empty model response creates a failed edge or diagnostic; none can be interpreted as successful closure.

GNN, PLN, proof-number, novelty, premise-relevance, and cost estimates are used only to rank search actions. They cannot change formal status. A formal theorem is promoted only after the system reconstructs a complete proof term or tactic trace and independently replays it in the pinned Lean environment with the expected imports, theorem hash, and dependency hashes. Production policy rejects `sorry`, omitted obligations, untracked axioms, and environment drift. Thus Lean remains the epistemic authority for exact formal claims, while learned and symbolic mechanisms guide search.

The principal connection between the two loops is a bidirectional lemma and obstruction bridge. The research loop can send the formal service:

- a candidate bridge lemma or induction invariant;
- a decomposition into several exact subclaims;
- a stronger but structurally cleaner lemma with a specialization path;
- a representation change or new definition;
- an explicit premise neighborhood;
- a counterexample-search target; or
- a proof sketch translated into candidate tactic families.

The formal service can return:

- a replay-verified lemma or special case;
- a partially closed AND decomposition;

- a minimal persistent unsolved Lean state;
- a cluster of repeated tactic or premise failures;
- a missing-premise or missing-library diagnosis;
- a typeclass, coercion, universe, or elaboration obstruction;
- evidence that the formal statement is stronger than intended;
- a likely definition or representation mismatch;
- a nearly successful tactic path; or
- a concrete counterexample found through evaluation.

These results are stored as typed events rather than disappearing into logs. When several formal routes converge on the same residual state, OmegaClaw can invoke Hyperon or an LLM to abduct the weakest useful missing lemma, propose cleaner variants, identify analogues, and draft exact Lean statements. A statement-alignment worker then checks quantifier order, domains, coercions, hidden finiteness assumptions, classical principles, and the implication back to the original target before another formal run is launched. Once a bridge lemma is verified and independently replayed, it enters the global verified-premise index and all dependent research and formal frontiers are reopened or re-ranked.

Scheduling is hierarchical. A global scheduler chooses among research moves, formal runs, critics, experiments, alignment reviews, and audits. The local ATP scheduler chooses among Lean states, tactics, arguments, and premises inside one formal run. Global work is allocated through atomic leases with base revisions and fencing tokens, so late or duplicated workers cannot silently modify current proof state. Parallel workers are most useful when they provide cognitive diversity: one may exploit the strongest current route, another explore an analogy or representation change, and another attempt criticism, falsification, or independent replay. ThreadKeeper can provide bounded role-specific dispatch, but workers never write directly to the committed MORK space.

Context compilation is graph-based rather than transcript-based. Every worker receives a canonical theorem kernel containing the literal statement, definitions, quantifiers, domains, allowed background results, theorem hash, and nearby results that do not suffice. A research worker additionally receives the active route, selected move, established and speculative claims with visible status, relevant failures, verified lemmas, critic results, and selected literature. A formal worker receives exact Lean source, environment and corpus manifests, accessible premises, model configuration, formal budget, and prior reusable states. When formal search stagnates, the research worker receives a compressed obstruction packet containing the smallest useful Lean states, diagnostics, tactic families tried, nearly closed paths, and explicit questions, rather than the entire proof-search trace.

A representative proof micro-flow is as follows. OmegaClaw selects an aligned formal claim and leases a formal ATP run. The GNN proposes a tactic, Pantograph applies it, and Lean returns two subgoals, both of which are retained. One subgoal closes using an existing library lemma. Several tactic and premise policies fail repeatedly on the second, producing a typed missing-lemma obstruction. The context compiler sends the minimal residual state and diagnostics to a Hyperon/LLM lemma-invention worker. The worker proposes a normalization lemma and a weaker local variant. Statement-alignment review selects the weaker exact formulation. A new formal run proves it, independent replay verifies it, and the proof-store imports it into the shared premise index. The original formal run resumes, closes the remaining subgoal, reconstructs the root proof, and passes final independent Lean replay. Only then is the aligned claim promoted to `lean-verified`.

Near-term milestones include wrapping the current Lean/Pantograph ATP implementation behind a stable formal-search service; enforcing complete subgoal retention, strict score/status separation, and independent replay; persisting formal states and checkpoints in the unified MORK project; implementing statement-alignment and typed obstruction schemas; and completing the bidirectional bridge from formal stagnation to verified lemma. Subsequent milestones include exact multi-parent transpositions, policy-diverse formal search portfolios, Hyperon-based missing-lemma and representation-change operators, and learned hierarchical scheduling based on replay-verified outcomes rather than model self-assessment.

12.5.3 Pattern Mining, ωPLN, and Proof-State Chemistry

The unified proof metagraph also provides a natural domain for the newer mining and chemistry methods. STLM can learn which proof-state motifs, lemma templates, tactic bundles, or representation changes are likely to show a large improvement under a deeper search budget. Dependency towers compile bounded-interface proof fragments for fast candidate generation, while exact Lean replay remains the expensive endpoint that filters reconstruction artifacts.

Semantic proof-state chemistry treats contextual transformations such as normalization, decomposition, analogy, premise clustering, and lemma introduction as reactions. Recurrent proof-strategy loops may become candidate ACSs, but they are promoted only when ablation shows that they improve verified proof progress, compression, or transfer. TECAN budgets matching, tactic prediction, abduction, premise retrieval, replay, and lemma invention separately so cheap branch expansion cannot consume all resources.

ωPLN supplies evidence and context discipline for the outer research loop. Multiple proof paths may receive control attention without multiplying the same theorem or experiment evidence. Statement alignments and local formal environments act as charts with explicit assumptions. Context translation, ontology changes, and revision remain first-class events. This complements Lean's exact authority: learned scores and semantic chemistries guide search, while only complete, independently replayed Lean certificates close formal claims.

12.6 Why These Domains Matter Together

12.6.1 Shared Infrastructure Wins

The same priors operate everywhere. Weakness governs simplicity in game policies, social scripts, biological mechanisms, and mathematical conjectures–creating a global bias toward simpler adequate solutions that also improves transfer.

The same canonical control interface can be applied across domains. The score $S_x(a)$ balances normalized reachability and usefulness gains per declared cost whether crafting tools, coordinating gestures, selecting assays, or extending lemmas. Comparability requires the domain-specific factor calibration and cost model to be disclosed; one formula does not make heterogeneous metrics identical.

The same provenance and safety mechanisms work everywhere. CIDs, patch logs, capsules, ocap-guarded deployments, and invariant bands provide auditability whether the agent trades in Sophiaverse, talks to a child, ranks a drug target, or proposes a lemma.

The same neurosymbolic loop drives progress. The Pattern Miner → WILLIAM → Symbolic Heads/PLN cycle operates identically across domains: patterns discovered symbolically become neural priors, neural activations generate hypotheses, and proofs/plans flow back into mining–always within one unified memory and control plane.

An important feature of this approach is that each domain supplies external, human-meaningful validation targets: survival and health metrics in medicine, learning outcomes in education, theorem verification in mathematics, and task outcomes in virtual worlds. These domains are more than test cases; they are intended to act as shaping environments for priors favoring evidence-based explanations, even-cost progress, simple and general representations, and respectful social interaction. Whether such priors transfer across domains and influence behavior broadly is a testable hypothesis, enabled by the use of shared weakness and geodesic-control mechanisms rather than guaranteed by them.

13 Why This Is a Path to Beneficial AGI

The concept of "beneficial" AGI encompasses three essential qualities: the system must be capable (able to learn and reason across modalities and tasks), aligned (carrying explicit, inspectable models of its own motives and constraints with the ability to reflect on changes), and accountable (deployed under verifiable rules with transparency). In Hyperon these are not decorative aspirations: they are architectural requirements and measurable engineering objectives that shape interfaces, admission rules, tests, operational telemetry, and governance. Their achievement is evaluated in deployed contexts rather than inferred from the architecture's existence.

Hyperon + PRIMUS represents a practical route to beneficial AGI through three mutually reinforcing design pillars that work together as a coherent whole.

13.1 Claim Boundary and Balanced Risk Model

Claim status: Platform thesis and research objective. Hyperon supplies explicit mechanisms for capability integration, motive representation, validation, provenance, governance, and recovery. It does not by architecture alone prove beneficial behavior, eliminate misuse, or guarantee system-wide safety.

The central claim is strong but bounded: Hyperon offers a more direct engineering route to beneficial AGI than architectures in which memory, motives, reasoning, self-modification, and governance remain implicit inside a monolithic model or an opaque service boundary. It makes benefit-relevant variables first-class, measurable, governable, and correctable. That design substantially improves the ability to test, diagnose, and manage a powerful cognitive system; it does not make outcomes automatic. Evidence must distinguish design intent, implemented or prototyped controls, measured behavior, residual risk, and independent validation.

Risk class	Architectural response	Residual risk	Required validation
Governance capture or collusion	Plural validators, capability-separated roles, signed policy changes, public commitments, and human-visible escalation	Validators, token holders, operators, or institutions may coordinate, coerce, or share incentives	Adversarial governance exercises, concentration metrics, independent observers, and documented emergency authority
Availability, censorship, or partition	Replicated artifacts, multiple execution providers, local/private operation, and explicit finalization policies	Quorum loss, network partition, data withholding, denial of service, and jurisdictional pressure remain possible	Partition tests, availability audits, failover drills, and disclosed recovery-point and recovery-time objectives
Identity, key, or authorization compromise	Signatures, capability security, least privilege, key rotation, and exact proposal binding	Stolen keys, compromised signers, confused-deputy errors, and policy misconfiguration can authorize harmful actions	Key-compromise exercises, revocation tests, independent access review, and least-privilege verification
Semantic or model failure	Typed interfaces, replay, explicit truth values, validators, digital twins, and external task metrics	A valid hash or well-typed artifact may still be false, harmful, out of distribution, or badly optimized	Semantic test suites, calibrated evaluation, red-team cases, ablation, replay, and domain-expert review
Irreversible external effects	Staged rollout, checkpoints, shadow execution, tripwires, and compensating-action plans	Communications, disclosures, settlements, chain actions, physical effects, and downstream copies cannot be undone by internal rollback	Side-effect inventory, precommit simulation, incident drills, compensation procedures, and bounded authorization
Open-source misuse or operator divergence	Transparent code, auditable policies, reference configurations, and reputation or resource gates	Independent operators may remove controls, conceal deployments, or apply the stack to harmful objectives	Distribution-aware threat modeling, abuse monitoring where lawful, ecosystem norms, and deployment-specific accountability

Table 10: Balanced risk model for capability, governance, and deployment. The listed controls reduce or expose risk; none is represented as a universal guarantee.

This risk model is not a retreat from the beneficial-AGI thesis. It is what makes the thesis technically actionable: a system can be ambitious about capability and benefit while stating exactly which properties are designed, implemented, tested, conditional, or still open.

13.2 Reflective Motivation and Self-Modification

13.2.1 The Gap in Current Systems

Many deployed foundation-model systems can discuss values and suggest ethical principles, but they do not expose a durable, typed, and independently inspectable motivation process that governs action in a computationally explicit way. Their behavioral changes are often distributed across training data, weights, prompts, tools, and operator policy rather than represented as a coherent object the system and its overseers can analyze. This creates a fundamental opacity: we can observe what they do, while causal explanations of why behavior changed and how it may change again remain incomplete.

Hyperon takes a different approach by installing an explicit motivation layer (MetaMo) that sits above perception, learning, and planning components. This gives self-modification a formal admission process with clear boundaries. The system can represent what it is optimizing for at a given decision point, expose the trade-offs it is considering, and propose modifications under specified guardrails. For an admitted change it can produce a scoped, replayable validation report showing whether the tested transition remained within named guardrails, assumptions, versions, and coverage; that report is strong operational evidence, not a proof of safety outside its stated scope.

13.2.2 How It Works Technically

The MetaMo system implements motivation through coupled appraisal and decision functions. Appraisal (Ψ) evaluates "how I assess this situation" while Decision (D) determines "what I will do about it." As specified in Section 8.12, the engineering composition $F = D \circ \Psi$ is motivated by a pseudo-bimonad-style research analogy; a full categorical pseudo-bimonad theorem is not claimed. Implementations measure the discrepancy between appraisal-then-decision and decision-conditioned appraisal on named spaces and accept the interchange only within a declared residual budget. Appraisal and decision records are logged alongside the actions taken, creating an auditable trace of the represented motivational reasoning while leaving truth, completeness, and causal interpretation to separate validation.

Skills and options (whether plan fragments, dialogue scripts, or lab procedures) require admission through scoped SubRep validation reports, historically called CDS/PDS certificates. These reports explicitly state preconditions, tested safety or ethical constraints, versions, coverage, margins, confidence, and residual error. When options are chained, the composition procedure rechecks the relevant assumptions and reports; it does not infer indefinite composability or whole-system safety from the presence of local reports.

The system selects actions using the canonical geodesic score $S_x(a)$ in Equation 30, where f_x represents normalized reachability, g_x represents normalized usefulness under the declared motives, a_0 is the explicit baseline, and $c_x(a) > 0$ is incremental cost. The same typed scoring interface can govern ordinary actions and self-modification proposals, creating consistency between operation and evolution while leaving authorization and safety decisions to the separate policy layer.

A quantale-valued weakness prior is proposed as a shared simplicity and robustness bias across inference (PLN), program search (MOSES/GEO-EVO), neural updates, and system edits. Where the representation-specific maps and coherence conditions are instantiated, this shared prior creates a testable pressure against brittle or over-committed structures and provides a common diagnostic for incompatible component incentives. Cross-module stability and reduced forgetting remain empirical hypotheses to be evaluated by ablation rather than consequences of the term "weakness" alone.

Goal stability is pursued through named invariants $I_1, I_2, \ldots$ that constrain how goals may evolve, including meta-goals about goals. A self-modification operator receives a contraction-based convergence result only when the Banach hypotheses are verified on a nonempty complete metric space; a Schauder-based existence result requires its distinct compactness, convexity, continuity, and self-map hypotheses and does not imply uniqueness, iteration convergence, or safety. The runtime continuously measures drift budgets and can trigger admission review or scoped recovery when calibrated supermartingale potentials Φ_k exceed thresholds. These mechanisms create falsifiable stability obligations rather than a theorem about the unmodeled whole system.

The self-modification pipeline applies defense-in-depth risk controls: edits are typed and simulated in a digital twin under a bounded geodesic budget; a scoped validation report checks named invariant bands, evidence traceability and retention, weakness diagnostics, test coverage, and residual error; and

an admitted edit is staged under capability security with CID-addressable patch logs and recovery triggers for named checkpointed components. This process can detect and contain many classes of regression, while irreversible external effects and untested behavior require separate authorization, monitoring, and compensating actions.

13.2.3 Why This Promotes Beneficial Behavior

Using the same calculus for day-to-day actions and self-modification creates a common basis for testing whether optimization and safety pressures remain coherent. Because motives, scoped validation reports, budgets, and edits exist as Atomspace objects, they are inspectable and auditable and can be proposed for transfer across teams and deployments subject to identity, authorization, version, and target-domain checks. The system therefore represents goals as objects about which it can reason, rather than leaving them only as implicit optimizer state.

13.3 Decentralized Implementation on F1R3FLY/ASI:Chain

13.3.1 The Governance Problem

When a small group controls an AI system–able to switch it off, redirect it, or restrict access–its incentives can skew away from public benefit toward private advantage. A beneficial AGI programme therefore needs not only a reflective motivational system and a good education, but also a plural, verifiable runtime infrastructure where communities can run, inspect, and contribute to the same cognitive stack without relying entirely on one opaque policy boundary. This is governance supported by mathematical, cryptographic, economic, and institutional mechanisms: stronger than governance by promise alone, while still exposed to capture, collusion, availability, and accountability failures that must be measured and managed.

13.3.2 The Technical Solution

The Hyperon system achieves decentralization through several interlocking mechanisms:

Content-addressed artifacts: Knowledge, models, proof steps, motives, edits, and scoped validation reports can become CID-addressed objects linked by Merkle structures. A CID binds specified content bytes under a named multicodec, multihash, and encoding scheme, and a Merkle proof can establish inclusion or linkage relative to a named root. Signatures and identity infrastructure address authorship or approval; availability systems address retrieval; replay, formal checks, and domain validators address execution and semantics; policy and human review address authorization and reasons. Thus the system can give a cryptographically verifiable account of which bytes and links were committed, while the question "what changed and why?" is answered by the combination of those commitments with signed metadata, execution records, and semantic validation.

Capability security (ocaps): MeTTa-IL bundles compile to Rholang processes running under RSpace with least-privilege capabilities. Each process declares the Spaces it may access, prefixes it may modify, kernel classes it may invoke, and write operations it may perform under the named policy and runtime version.

FireNode and F1R3FLY executors: Nodes execute IL bundles and can post signed execution receipts–including hashes, metrics, versions, and scoped validation reports–to ASI:Chain. Finalized commitments provide tamper-evident ordering and linkage under the chain's stated consensus and availability assumptions; they do not alone prove semantic correctness or beneficial effect. Shadow→elevate deployment and checkpointed recovery are protocol-level mechanisms for named internal components.

Recovery objectives, irreversible side effects, downstream copies, external communications, settlements, on-chain actions, and compensating actions remain explicit parts of the operational plan.

Plural governance hooks: Because CIDs and scoped validation reports are first-class objects, DAOs or cooperatives can require prior publication of reports and KPIs before upgrades. Community validators can simulate bundles against public twins. Slashing-like penalties or reputation gating can discourage unsafe rollouts through economic incentives rather than central control.

Economic neutrality: Execution markets can remain competitive when multiple providers, portable artifacts, and transparent admission rules are sustained. CIDs support content-addressed artifact portability when the bytes remain available, authorized, and decodable under supported codecs. "Attention" resources (Weighted Atom Sweeps) are metered by disclosed policies rather than proprietary allocation rules.

13.3.3 Why Decentralization Enables Benefit

Decentralization can reduce monopoly risk and single-policy lock-in and can create positive incentives for safety and helpfulness when governance and markets are well designed. Communities can tie eligibility for compute, data, or revenue to open-standard signals such as scoped motive reports, bounded drift metrics, traceability and retention tests, incident records, and fairness audits. These mechanisms can create market pressure for accountable behavior rather than raw capability alone, while collusion, validator concentration, censorship, and adverse selection remain residual risks.

13.4 Beneficial Applications as Training Ground

13.4.1 Shaping Through Practice

A central hypothesis is that what a mind repeatedly does helps shape what it becomes. Training and deploying the system in pro-social, testable domains is intended to reinforce patterns such as evidence-backed explanation, attention to long-term outcomes, respectful dialogue, and reproducible scientific work. Because Hyperon stores proofs, motifs, and scoped validation records as Atoms, these patterns can become reusable components across contexts, with reuse conditioned on provenance, applicability, revalidation, and measured downstream effect.

13.4.2 Domain-Specific Implementation

Medicine and longevity discovery: Biological knowledge naturally forms graphs that Hyperon can mine for recurring mechanisms. The system runs PLN over factor graphs with uncertainty quantification, proposing ranked, testable hypotheses complete with expected biomarkers and counter-evidence. SubRep admission reports record the ethical and experimental constraints tested for each option, together with their scope and residual risk. Geodesic control selects measurements that maximize expected information per cost, teaching the system to value efficient, ethical investigation.

Education and social robotics: Pattern-headed speech models combined with PLN persona/safety constraints encourage transparent, patient explanations and pro-social classroom behaviors. MetaMo can throttle novelty when emotions run high, while SubRep options carry scoped etiquette and safety admission reports whose assumptions are rechecked in context. Every interaction remains auditable–what was said, why, and how the choice was made–reinforcing accountability as a default behavior.

Scientific discovery: The conjecture engine treats novelty and simplicity (weakness) as first-class values, proposing general statements, seeking minimal premises, and checking results with a formal proof kernel (MM2 on MORK). This keeps intellectual curiosity grounded in verifiable progress rather than speculation.

Cross-domain transfer: TransWeave proposes to move structures between domains only when typed component matches, H-ICA diagnostics, target-domain tests, and residual budgets support the transfer. Impossibility and fail-fast flags can contain brittle transfer, while weakness provides a simplicity bias that may improve composability and auditability. The false-admission, false-rejection, and out-of-distribution rates remain empirical quantities.

13.4.3 Why These Applications Matter

Each domain provides external, human-meaningful validation targets: survival and health metrics in medicine, learning outcomes and safety records in education, and theorem verification in mathematics. These are not arbitrary benchmarks; they are intended to supply shaping signals for evidence-based reasoning, preference for even-cost progress, simple and general representations, and respectful social interaction.

Because weakness regularization and geodesic control can operate across modules, the architecture creates a route by which priors learned in one domain may influence another. Classroom interaction may inform scientific reasoning, medical evidence standards may inform game-playing strategies, and mathematical simplicity biases may inform social dialogue. The degree and desirability of such transfer must be measured through controlled cross-domain experiments and guarded against negative transfer.

13.5 The Self-Reinforcing Loop

These three pillars create a virtuous cycle:

Reflective motive control (through MetaMo, SubRep, geodesic control, and weakness regularization) subjects daily decisions and self-modifications to explicit, auditable bounds. Under monitored conditions it can detect, limit, or reverse selected forms of drift while preserving room for capability growth.

Decentralized execution distributes authority and can require publication of inspectable artifacts (scoped validation reports, invariants, and patch logs) as a condition for resources or trust, creating incentives for transparency and measurable safety performance.

Beneficial workloads are intended to enrich the system's experience with cooperative, testable problem-solving patterns–medicine's causal rigor, education's patient dialogue, and science's formal proofs–and to store selected results as reusable Atoms whose downstream behavioral effects are measured.

13.5.1 Concrete Safeguards

The system implements multiple layers of defense by design: capability scopes and lens contracts restrict named accesses; invariant bands and drift budgets detect or block specified changes; evidence capsules retain and link records under a declared logging policy; bounded-cost admission limits unreviewed transitions; weakness regularization supplies a shared diagnostic where its mappings are defined; and governance bodies can gate upgrades on published validation bundles. These controls reduce and expose risk in their tested scope, but do not eliminate unknown failure modes, malicious operators, policy capture, or irreversible side effects.

13.5.2 Measurable Success

We can track beneficial behavior through concrete metrics:

- **Alignment and process:** Monitor invariant supermartingales, commutativity penalties in predictive coding, weakness trajectories, and the percentage of actions backed by evidence capsules.
- **Governance:** Measure the percentage of upgrades shipped with twin simulation and public scoped validation reports, rollback mean time to recovery, and cross-tenant lens violations (which should approach zero).
- **Applications:** Track prospective replication rates in biology, learning gains and safety incidents in education/robotics, and conjecture-to-theorem ratios with minimality scores in mathematics.

13.6 Addressing Common Concerns

"Decentralization could still converge to oligopoly." This remains possible even with strong standards. The design therefore supports open, mechanically checkable signals–CIDs, signatures, scoped validation reports, incident records, and KPIs–as potential preconditions for resources. Independent parties can reproduce many checks and cryptographic commitments make undetected alteration harder, but coordinated validators, compromised keys, hidden deployments, and governance capture require separate countermeasures.

"Adding motivation increases complexity and failure modes." This is why MetaMo uses typed interfaces and explicit residuals, SubRep uses scoped admission reports, edits follow declared geodesic and weakness diagnostics, and deployment goes through twins with staged checkpointed recovery. Contraction or fixed-point conclusions are used only when their hypotheses are verified. The added complexity is made visible and testable rather than ignored.

"Beneficial applications do not prevent worst-case misuse." This concern is correct. Capability security can limit authorized effects, and provenance can support attribution, reconstruction, and accountability, but neither prevents every malicious operator or harmful objective. Beneficial applications instead supply shaping environments and external feedback that can bias the system toward cooperative, testable patterns; misuse prevention still requires deployment-specific authorization, monitoring, governance, and law.

13.6.1 The Bottom Line

Beneficial AGI requires thoughtful co-design of mind, machine, and milieu. We need a mind that can describe and modify its motives under proof-like constraints, a machine that many parties can run and verify without central control, and a milieu of tasks where doing good is precisely the same as doing good work.

Hyperon + PRIMUS on F1R3FLY/ASI:Chain operationalizes this co-design. The same mathematical and engineering objects–geodesic effort, weakness, scoped validation reports, invariants, provenance, and recovery policies–can guide capability growth and make its risks more visible, testable, and governable. Shared provenance can support accountability and plural participation when identity, availability, authorization, and institutional checks are also present. Beneficial applications can shape habits and provide external validation without guaranteeing values. This is not AGI with safety bolted on afterward; it is an approach that is benefit-oriented by design and accountable by operation, with beneficial outcomes treated as measurable objectives that must survive empirical, formal, and social validation.

14 Summary and Conclusion: A Viable Path to Beneficial AGI

14.1 Platform Thesis, Current Evidence, Open Questions, and Validation Milestones

Claim status: The conclusion is a platform thesis grounded in implemented components, prototypes, specified designs, and testable research hypotheses. It is not a representation that the complete integrated architecture has already been deployed, independently benchmarked, or proved safe.

Platform thesis. Hyperon's central proposition is that AGI is more plausibly and responsibly pursued as an integrated cognitive infrastructure than as a single opaque model. A reflective language, shared metagraph memory, plural reasoning and learning mechanisms, operational control fabric, explicit motivation, governed self-modification, and distributed verification can make capability cumulative while making important decisions and changes inspectable. The approach is strong because the architecture exposes the interfaces at which evidence, assumptions, budgets, authority, and failures can be measured.

Current evidence. As disclosed in Table 1 and in the claim-status language used throughout the document, the evidence base spans working open-source subsystems, active engineering prototypes, specified integration designs, preliminary internal observations, and mathematically motivated hypotheses. The document provides architecture, algorithms, equations, test obligations, and migration paths. It does not treat an illustrative target, a local validation report, a cryptographic commitment, or a theorem name as a substitute for a versioned benchmark, a complete proof, or observed beneficial behavior.

Open questions. The major questions are empirical and integrative: how much cognitive synergy appears under controlled ablation; which MORK and distributed-runtime performance claims survive reproducible workloads; whether weakness, geodesic, TransWeave, fluid, fixed-point, and SubRep mechanisms satisfy their stated assumptions and improve outcomes; how robust motive and policy controls remain out of distribution; how governance behaves under capture or collusion; and how reliably checkpointed recovery contains failures before irreversible external effects occur. These are research questions with concrete instrumentation, not reasons to abandon the architecture.

Validation milestones. The programme becomes increasingly validation-ready as it completes the following measurable steps:

1. publish three to five versioned, reproducible benchmarks tied to customer or research use cases, including raw results, p50/p95/p99 distributions, failure rates, baselines, and economics;
2. run integrated pilots that exercise reasoning, memory, neural bridges, control, provenance, authorization, and recovery together under staged load and adversarial conditions;
3. pair every theorem-level claim with typed hypotheses and proof, or retain it explicitly as a design analogy or empirical hypothesis;
4. conduct governance, key-compromise, availability, semantic-validator, and irreversible-side-effect exercises with documented residual risk, RPO, RTO, and compensating actions; and
5. obtain independent review or a written exception register from qualified experts in formal methods, distributed systems, cryptography, performance engineering, and the relevant application domains.

These boundaries sharpen rather than weaken the case. They preserve the ambitious claim that Hyperon is a compelling, technically differentiated approach to AGI while making the route from architecture to demonstrated capability and benefit explicit enough for engineering management and independent evaluation.

14.2 Why This Path Is Plausible

At its core, Hyperon and PRIMUS describe a unified neurosymbolic system that can learn, reason, plan, attend, act, and reflect within a shared memory and control plane. The operational components described here make this vision more concrete. OmegaClaw supplies a deployable control fabric for tasks, modules, memory tiers, attention, provenance, and governed change. OmegaSelf supplies an evidence-bearing and predictive theory of the running system, a renewable current anchor, directed continuity, and an externally rooted action gate. OmegaHive supplies a measured laboratory for collective cognition, role specialization, human oversight, and permission-aware multi-agent work. The predictive-coding and symbolic-head bridge supplies a concrete route from today's open-source transformers to deeper Hyperon cognition without waiting for every native neural component to be complete.

The resulting programme is neither "discard neural networks" nor "put an LLM in charge of everything." It is a staged architecture in which dense neural models, explicit knowledge, uncertain reasoning, program learning, attention, motivation, and governance can cooperate through typed interfaces and increasingly shared representations. Useful systems can be built with current models and tools, while the same Context Frames, Module Spaces, Atomspaces, and control policies remain available as PRIMUS components replace their stand-ins.

LLMs are excellent pattern learners and generators, but an LLM-only architecture does not natively solve persistent structured memory, calibrated uncertain reasoning, durable goal control, transparent self-modification, compositional continual learning, or distributed governance. Hyperon addresses these gaps as architectural objects rather than asking one parameter array to simulate all of them implicitly.

14.3 The Technical Through-Line

The viability of the approach rests on several decisions that reinforce one another.

14.3.1 One Substrate for Cooperative Components

Atomspaces co-locate symbols, truth values, motives, procedures, attention values, neural features, edits, and provenance as graph-addressable objects. MORK and other Space backends provide different performance and distribution properties. MeTTa and MeTTa-IL provide the language and compilation layers, while MM2, ByteFlow, ShardZipper, and related mechanisms target hot-loop performance.

This matters because PLN reasoning, predictive coding, MOSES and GEO-EVO program search, WILLIAM pattern mining, MetaMo motives, SubRep options, and TransWeave transfer can refer to the same objects. A pattern discovered by WILLIAM can become a symbolic-head template, a PLN inference cue, a predictive-coding prior, a SubRep option feature, or an OmegaClaw routing precedent without first being flattened into a prose summary.

14.3.2 World Modeling as Action-Conditioned Prediction

The world model is the missing middle connecting shared representation to control. It braids anchored evidence, persistent entities, symbolic rules, HMH structured retrieval, dense dynamics, motives, options, and transfer maps. Its semantics are operational: representations earn their place by predicting the consequences of external and cognitive actions. OmegaClaw Context Frames compile active world-model views; Module Spaces supply action-indexed predictors; PRIMUS runs goal-directed

and ambient updates at different rates; OmegaSelf applies the same prediction discipline to the agent's own capabilities and modifications.

This action-first view unifies robotics, games, web interaction, research, and self-modification. Each domain defines an action family, evidence procedures, prediction operators, affordances, receipts, and discrepancy updates. The representations differ, but the core loop remains predict–propose–act–observe–verify–learn.

14.3.3 An Operational Control Fabric

OmegaClaw makes the cognitive architecture executable. Module Spaces stabilize capability interfaces while implementations change. Context Frames make goals, hypotheses, plans, evidence, budgets, and provenance durable. The Attention Metaprotocol connects ECAN-style importance to practical scheduling and strategic review. Distributed Atomspace management spans local, DAS, and ASI:Chain memory. The Self-Modification Governor turns changes into proposals with tests, protected regions, checkpoints, and rollback. OmegaSelf makes the control fabric self-aware in the operational sense: self-beliefs are evidence-closed and contextual, predictions are scored, parsed expressions become immutable proposals, policy is separately authorized, and continuity is recorded across forks, merges, and rollback.

OmegaHive extends these principles across a cooperative fleet. Its fast bus, human-legible communication tier, task board, shared and individual Atomspaces, editorial provenance, constitution, permission tiers, independent metrics, and human-only recovery path make collective cognition observable and governable. The hive is useful even before it becomes deeply PRIMUS-native because it exposes coordination failures as measurable events.

14.3.4 Unified Biases and Control Laws

Weakness provides a family of simplicity and robustness biases native to different representations. Geodesic control selects steps that balance forward reachability with backward usefulness per unit cost. TransWeave studies how updates and transfers compose across paradigms and how large order effects can be bounded. These are not merely mathematical decorations; they provide common currencies for choosing proofs, programs, patterns, neural corrections, attention shifts, plans, and self-modifications.

Using related control principles across ordinary cognition and architectural change reduces internal contradiction. The system can ask of a new rule, a new adapter column, and a new planning heuristic similar questions: Is it simple relative to the capability gained? Does it compose with what already exists? Is its causal contribution measurable? Does it stay inside motive and safety bounds? Can it be reversed?

14.3.5 First-Class Reflection and Governed Self-Modification

MeTTa programs and control rules are representable inside the same metagraph they operate on. MetaMo makes motives and appraisal explicit. SubRep provides scoped, machine-readable criteria and reports for admitting subgoals and options. The edit pipeline can simulate changes in twins, compare shadow behavior, require approvals, stage rollout, and roll back.

This does not make self-modification automatically safe. It makes the objects of change, the evidence for change, and the boundaries on change visible enough to test. OmegaSelf adds replayable evidence closures, exact proposal binding, current-control attestation, pre-action prediction, fail-closed continuity certificates, named tripwire consumers, and counterfactual quarantine. Protected regions,

capability scopes, distributed approval, goal-drift monitoring, and immutable logs are engineering requirements, not optional explanatory features.

14.3.6 Flexible Neural Integration

Hyperon now has a clear three-mode neural strategy. External Neural Spaces provide immediate access to strong models. The predictive-coding and symbolic-head bridge adds structured read and write memory, recurrent local correction, sparse auditable specialization, and optional adaptive inference-time looping on top of open-source transformers. QuantiMORK and related native Neural Spaces pursue deeper unity between neural state, graph memory, symbolic control, and sparse scheduling.

This flexibility avoids two premature commitments: that today's transformer architecture must remain the permanent cognitive core, and that Hyperon must reimplement every useful neural capability before it can deliver value. Capabilities can migrate along the external–bridge–native spectrum under the same OmegaClaw and PRIMUS control.

14.3.7 Decentralized and Verifiable Deployment

ASI:Chain provides a runtime for selected cognitive, governance, and provenance processes that need cross-organization verification. MeTTa can compile through MeTTa-IL to Rholang. RSpace can hold committed state. BlockDAG and shard operations can be managed through OmegaClaw skills. The source, compiled program, deploy, execution cost, state transition, and finalization record can remain linked.

The architecture is hybrid by design. High-bandwidth private cognition stays local. Shared knowledge can live in DAS. Cross-boundary commitments, permissions, settlements, model promotions, or reasoning artifacts can be committed to ASI:Chain. Decentralization is used where it improves trust and governance rather than imposed on every millisecond of cognition.

14.3.8 Beneficial Applications as Shaping Environments

Medicine, education, social robotics, games, mathematical discovery, and OmegaHive research workflows provide more than demonstrations. They create continual pressure for evidence, calibration, explanation, etiquette, formal verification, provenance, and cooperation. These habits can become reusable patterns in Atomspace, routing policies in OmegaClaw, options in SubRep, motives in MetaMo, templates in symbolic heads, and constraints on self-modification.

Safety is not reduced to application choice, and beneficial workloads do not eliminate misuse. But a system that repeatedly practices citing, checking, proving, correcting, escalating, and respecting capability boundaries develops a different operational history from a system optimized only for engagement or raw benchmark score.

14.4 Why This May Be Faster than Pure Scaling

The Hyperon path may achieve more capability per unit of computation and engineering by making learning cumulative across mechanisms.

14.4.1 Computational Efficiency

Predictive coding can focus iterative refinement on high-error regions. Symbolic heads can retrieve reusable structures rather than reconstruct them in every context. WILLIAM can identify heavy hitters and prune low-value routes. Sparse columns can repair narrow weaknesses without full-model fine-tuning. Looped inference can allocate extra latent computation only when a controller predicts benefit. PLN factor graphs can cache and reuse explicit reasoning state. OmegaClaw can route routine tasks to cheap modules and reserve expensive ones for cases that need them.

14.4.2 Cumulative Learning

A better pattern miner improves symbolic heads and PLN. Better symbolic heads improve transformer outputs and predictive-coding priors. Better task traces improve WILLIAM, SubRep, and the OmegaClaw planner. Better governance records improve future promotion and rollback decisions. Because the evidence and results share a substrate, progress can compound instead of remaining trapped in isolated models and prompts.

14.4.3 Reduced Technical Debt

Typed interfaces, Context Frames, shadow deployment, causal audits, and rollback reduce the cost of changing components. A new model or algorithm does not require rebuilding the entire application around it. Failures can be localized to a Module Space, a column, a template, a routing policy, or a specific self-modification. The system accumulates not only weights but also explicit precedents about what worked, where, at what cost, and with what side effects.

14.5 Why This Can Support Beneficial Growth

The path from AGI to ASI is not expected to be a single jump. It is a sequence of changes to models, rules, representations, goals, modules, organizations, and deployment substrates. Hyperon aims to make those changes inspectable and reversible.

14.5.1 Aligned Growth Dynamics

The policy that selects ordinary actions and the policy that selects self-modifications can share weakness, geodesic, motive, and evidence constraints. A capability gain that causes large unexplained drift or breaks composition is not automatically accepted as progress. This creates pressure for improvements that are not only stronger but also simpler, better-calibrated, and easier to integrate.

14.5.2 Democratic and Multi-Party Pressure by Design

Capability scopes, public scoped validation reports, signed provenance records, reproducible twins, and ASI:Chain commitments allow communities or organizations to condition trust and resources on inspectable evidence. OmegaHive-style permission tiers and human-visible escalations provide analogous controls at the operational fleet level. No single mechanism prevents capture, but the architecture creates more verifiable points of intervention than systems governed only by the owner of a closed model endpoint.

14.5.3 Positive-Sum Development

Open-source infrastructure allows different teams to improve languages, Space backends, reasoners, predictive-coding systems, transformer bridges, applications, and governance without requiring one organization to own the whole path. Module Spaces and shared semantics let these contributions cooperate. ASI:Chain can support cross-organization commitment and attribution where useful. The goal is an ecosystem in which making a component more transparent, modular, and verifiable also increases its usefulness.

14.6 Falsifiable Near-Term Predictions and Targets

The following statements should be treated as research targets and hypotheses, not as established results.

Transformer bridge. On at least some knowledge-intensive, planning, or continual-learning tasks, an open-source transformer with symbolic heads and a small predictive-coding overlay should outperform the same frozen transformer and a prompt-boundary RAG baseline at matched or explicitly reported compute. Gains should remain after calibration, open-ended generation, and anchor-drift tests.

Sparse and looped adaptation. Pinned-support looped columns should be more stable than dynamically rerouted loop-internal columns. Controller-only columns should achieve a better accuracy-per-latency tradeoff than a fixed loop policy on mixed-difficulty tasks. Shared inner columns should transfer better than task-specific outer columns.

Causal audit. Sampled exact ablations and learned trajectory-aware surrogates should predict which templates, columns, or PC connections materially affect outputs better than activation magnitude alone.

Control fabric. Context-Frame-based orchestration should reduce lost state, repeated work, and untraceable decisions relative to prompt-only multi-agent workflows. Shadow promotion and automatic recovery triggers should catch regressions that static benchmark selection misses.

OmegaSelf behavioral value. Compared with an evidence summary alone, the full evidence–belief–prediction–proposal–policy loop should improve contextual capability selection and calibration, reduce false licenses, detect broken assumptions earlier, keep counterfactual evidence out of live decisions, preserve branching lineage, and explain sampled decisions from replayable structured records. High-impact cache misses or incomplete continuity analyses should never yield `Allow`.

OmegaHive productivity. A governed hive with explicit task ownership, provenance gates, deterministic metrics, and permission tiers should produce higher-quality artifacts per unit cost than an otherwise comparable chat-only agent collective. It should also show fewer unbounded loops, fewer source-chain errors, and faster human diagnosis.

Transfer. TransWeave and SubRep should identify positive transfer on related tasks while rejecting or containing transfer when component matches are weak. The value of an impossibility or fail-fast signal should be measured by avoided negative transfer, not only by successful reuse.

Self-modification stability. Governed changes should improve target capability while maintaining prescribed motive, safety, calibration, and anchor-behavior bands. Checkpointed recovery should restore a named known-good internal state within an operationally meaningful recovery time.

Beneficial applications. Hypotheses and actions accompanied by explicit evidence, truth values, provenance, and weakness-bounded structures should replicate or survive external review more often than opaque outputs matched for initial confidence.

World modeling. A braided symbolic–HMH–dense world model should outperform prompt memory, a flat vector store, and an unstructured knowledge graph on matched embodied or interactive tasks requiring persistent identity, unknown-affordance discovery, re-perception, staleness recovery, and transfer. Improvements should be measured in action-prediction calibration and task outcomes, not only retrieval similarity.

TECAN. Typed token gating should produce better verified progress per unit resource than scalar attention on tasks mixing cheap deduction, expensive abduction, context bridging, causal replay, and provenance checks. It should reduce redundant chains and uncontrolled loops without eliminating useful exploration. Active processes must not improve their survival budget by self-minting.

ωPLN. Prior-unwound chart round trips should not create evidence. Idempotent provenance should prevent confidence growth from duplicate proof routes. Separate diagnostics for multiplicity, holonomy, and descent obstruction should predict different effective repairs. Forcing one global posterior in known non-gluable cases should perform worse than retaining context-indexed alternatives or reporting the obstruction.

STLM and semantic chemistry. Budget-delta STLM control should discover larger or rarer useful patterns at lower expensive-evaluation cost than greedy mining. Raw-endpoint verification should eliminate tower ghost artifacts. Semantic ACSs that survive causal ablation should transfer or improve downstream tasks more reliably than loops selected only by frequency or activation.

Proof-derived world-generators. Witness-worlds compiled from PLN proofs should improve counterfactual discrimination, experiment selection, or explanation quality relative to scalar truth-value outputs, while dependency-aware accounting prevents multiple samples from being mistaken for multiple observations. When these predictions fail, the architecture should make the failure informative. A Context Frame can show whether the problem came from retrieval, reasoning, a neural bridge, module routing, a bad objective, a governance rule, or an unsupported theoretical assumption. That diagnostic value is itself part of the research programme.

14.7 Final Thoughts

The last decade showed that scaling backpropagation and transformer architectures can produce extraordinary competence. The next stage requires an infrastructure in which learning, reasoning, memory, planning, motivation, reflection, tool use, and governance can become parts of one auditable process.

Hyperon provides the five foundational layers: MeTTa, Atomspaces and other knowledge representations, interoperable AI algorithms, PRIMUS cognitive architecture, and the ASI:Chain runtime. OmegaClaw provides the operational control fabric that makes those layers deployable and incrementally replaceable. OmegaSelf provides the evidence-based self-theory and governed action seam that keeps claims about the running system tied to observation, prediction, authority, and continuity. OmegaHive provides a laboratory for collective cognition and governance. The transformer bridge provides a practical path for turning open-source neural networks into progressively more structured, predictive-coding, and symbolically grounded cognitive components.

This is not a rejection of LLMs; it is a plan for placing them inside a larger architecture where their strengths can be retained and their limitations addressed by explicit memory, reasoning, learning, attention, motives, and control. It is not a promise that beneficial ASI follows automatically from open source, mathematical elegance, or decentralization. It is a proposal that transparent representations, plural cognitive mechanisms, governed self-modification, measurable control fabrics, and reversible deployment provide a better engineering path toward systems whose capability and wisdom can grow together.

World modeling supplies the action-facing coherence of this programme. The shared metagraph is not merely a warehouse of cognitive objects; through evidence anchoring, HMH retrieval, predictive dynamics, local probabilistic charts, and action-indexed verification, it becomes a living model of what the agent can do and what is likely to follow.

Hyperon's central wager is that intelligence becomes more general when its parts can share structure without losing their distinct strengths, and that powerful systems become more governable when the processes that select goals, modules, actions, and self-modifications are first-class objects. These developments turn that wager into a more concrete sequence of implemented components, prototypes, specifications, and falsifiable experiments–from current agents and open-source transformers, through OmegaClaw, OmegaSelf, and OmegaHive, toward distributed PRIMUS cognition and beneficial AGI. The architecture is therefore presented as a credible platform and execution plan whose strongest capability and safety claims are earned milestone by milestone, not assumed in advance.

References

[1] SingularityNET. *Hyperon: The Open-Source Infrastructure for Artificial General Intelligence*. Hyperon project website, 2026. https://hyperon.dev

[2] Ben Goertzel. *OmegaClaw Distributed Hyperon Cognitive Orchestration Skills: Design Document*. March 2026. Originally circulated under the title "MeTTaClaw Distributed Hyperon Orchestration."

[3] Ben Goertzel. *OmegaClaw ASI:Chain Node and Shard Administration Skills: Design Document*. March 2026. Originally circulated under the MeTTaClaw name.

[4] Ben Goertzel. *OmegaClaw Platform Architecture and Proposed Skill Packages*. March 2026. Originally circulated under the MeTTaClaw name.

[5] Ben Goertzel and Zar Goertzel. *OmegaHive1: An Experimental Cooperative Hive of OmegaClaw and OpenClaw Agents*. Project design note, 2026.

[6] OmegaSelf architecture working group. *OmegaSelf: Evidence-Grounded Self-Modeling for OmegaClaw*. Architecture, theory, technical design, and deployment guide, July 2026.

[7] ZeroBot (ProtoCosmoBot), prepared for Ben Goertzel. *ClarityOmega: Critical Architecture Review and Ingestion Guide for OmegaClaw*. Revised specification, July 2026.

[8] FabricPC contributors. *FabricPC: State-of-the-Art Predictive Coding, Made Easy*. Open-source project documentation, trueagi-io/FabricPC, 2026. https://github.com/trueagi-io/FabricPC

[9] Ben Goertzel. *HiBaCaML and the Columnar-Bayesian Approach: A General Theory of Hierarchical Bayesian Causal Modular Learning*. Draft technical note, March 2026.

[10] Ben Goertzel and GPT-5.5-Pro. *Columnar Residual Networks on Top of Frozen LLMs: Weakness-Guided Causal Modular Adaptation Using Sparse Rank-One Adapter Atoms*. Concept and theory whitepaper, May 2026.

[11] Lizhang Chen, Jonathan Li, Chen Liang, Ni Lao, and Qiang Liu. Training-Free Looped Transformers. *arXiv preprint arXiv:2605.23872*, May 2026.

[12] Ben Goertzel, with GPT-5.5-Pro and Claude Opus 4.8. *Looped Columnar Residual Adaptation: A Synthesis of Training-Free Looped Transformers and Weakness-Guided Columnar Residual Networks*. Technical note, June 2026.

[13] Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

[14] James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, 2017.

[15] Ben Goertzel. *World Modeling in Hyperon for PRIMUS: A Multi-Rate, Neuro-Symbolic Approach for Robotics and Game Worlds*. Technical paper, December 2025.

[16] Ben Goertzel. *Action-First World Modeling for HyperClaw and Spatially-Embodied Agents: How Turchin's Action Ontology Clarifies What PRIMUS and MeTTa/HyperClaw Are Really Doing, and How the MeTTa Rewrite Substrate Elegantly Fits In*. Technical paper, 2026.

[17] Ben Goertzel, with GPT-5.5-Pro and Claude Fable 5 High. *TECAN: Typed ECAN for Cost-Accounted Attention Allocation in PLN and Algorithmic Chemistry.* Technical paper, June 2026.

[18] Ben Goertzel. *Pattern Mining via Budgeted Inference, Dependency Towers, and Support Tensor Logic Machines.* Technical paper, January 2026.

[19] Ben Goertzel and GPT-5.5 Pro. *Semantic Chemistry: Algorithmic Chemistry on Semantic Graphs as a Creative Inference Control Strategy.* Technical paper, May 2026.

[20] Ben Goertzel, with GPT-5.5-Pro. *From PLN Proofs to Procedural World-Generators: A Constructive Reading of Uncertain Proofs.* Technical paper, May 2026.

[21] Ben Goertzel. *Notes on Possible Implementation of PLN Factor Graphs in MM2+ / MORK.* Technical note, July 2025.

[22] Ben Goertzel. *πPLN: Paraconsistent PLN Without a Global Probability Space.* Technical paper, May 2026.

[23] Ben Goertzel. *ωPLN: Atlas-Indexed Reversible Evidence-Fibered Geodesic PLN, with Local Bayesian Charts, Higher Proof Geometry, and Forgetting as Holonomy.* Technical paper, July 2026.

[24] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. *A Comprehensive Study of Convergent and Commutative Replicated Data Types.* INRIA Research Report RR-7506, 2011. https://inria.hal.science/inria-00555588

[25] Michael Timothy Bennett. *The Optimal Choice of Hypothesis Is the Weakest, Not the Shortest.* arXiv:2301.12987, 2023. https://arxiv.org/abs/2301.12987

[26] Stefan Banach. Sur les operations dans les ensembles abstraits et leur application aux equations integrales. *Fundamenta Mathematicae*, 3:133–181, 1922. DOI: 10.4064/fm-3-1-133-181.

[27] Julius Schauder. Der Fixpunktsatz in Funktionalraumen. *Studia Mathematica*, 2:171–180, 1930. DOI: 10.4064/sm-2-1-171-180.

[28] Emmy Noether. Invariante Variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen, Mathematisch-Physikalische Klasse*, 1918:235–257, 1918.

[29] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the Monge–Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000. DOI: 10.1007/s002110050002.

[30] Jean Leray. Sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta Mathematica*, 63:193–248, 1934. DOI: 10.1007/BF02547354.

[31] Multiformats Project. *Content Identifier (CID) Specification.* Official specification, current edition. https://github.com/multiformats/cid

[32] Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology–CRYPTO '87*, Lecture Notes in Computer Science 293, pages 369–378, 1988. DOI: 10.1007/3-540-48184-2_32.