FUTON0

@multiarg futon0/devmap

@title FUTON0 Development Map — Human-System Interface

@audience futon-devs, reflective-agents, future-you

@tone formal-analytic

@style roadmap

@factor Mindful apprehension (sati + sampajañña)

@IFR: FUTON0 is the interface through which the system extends human capability. It monitors vitality, rhythm, and attention, making human state legible to the stack and stack state perceivable to the human. The human is not a component of the system; the system is an extension of the human.

@state P0 (core infrastructure) operational: Stack HUD, bidirectional channels, infrastructure substrate working. P1-P7 (layer interfaces) not yet established.

@next Establish F0-F1 interface (memory access), then F0-F2 (agent perception).

The Argument

FUTON0 is the interface through which the system extends human capability to reason, remember, verify, compose, and act effectively. The Stack HUD, voice channels, and infrastructure substrate establish that the interface works at all (P0). The remaining work is to extend this interface across each layer of the stack: F1 (memory), F2 (agents), F3 (verification), F4 (hypertext), F5 (patterns), F6 (mathematics), F7 (transparency). When all layer interfaces work, F0 is complete — the human can access the full capability of the system through a unified interface. Each layer reflects the whole from its perspective; F0 sees vitality, rhythm, and attention as the ground from which all other capabilities become accessible.

! instantiated-by: Prototype 0 — Core Infrastructure [🙇/亏 🔃/叉]

```
:maturity :active
:depends-on []
:evidence-for-active [Stack HUD operational;
bidirectional channels demonstrated;
infrastructure working]
```

+context:The interface must work at all before it can extend human capability.

+if:The human needs to perceive stack state and the stack needs to perceive human vitality

+however:These capabilities require working infrastructure, channels, and monitoring

+then:Establish Stack HUD, bidirectional channels (voice, keyboard, TTS), infrastructure substrate, health monitoring, affect signals, epistemic rhythm, and stewardship practices

+because:This is the foundation — without it, no layer interface can function

+evidence: evidence[Stack HUD (futon0/contrib/stack-hud.el) displays vitality + stack state] evidence[Voice-to-text, text-to-speech, Kinesis board operational] evidence[Infrastructure substrate (Alacritty, Emacs, Linode, FUTON1, Codex, Claude) working] evidence[Vitality scanner captures commit times and session timestamps]

+ NEXT-STEPS:
next[Complete health monitoring (sleep inference, energy inference)] next[Complete affect signal extraction from logged sessions] next[Complete epistemic rhythm checks (weekly/monthly)] next[Document stewardship protocol]

`:success-criteria`
pass[HUD displays vitality + stack state within 2 seconds] pass[Bidirectional channels feel unified, not siloed] pass[Health/affect/rhythm signals are meaningful, not noise] fail[Core infrastructure requires constant manual intervention]

! instantiated-by: Prototype 1 — F0-F1 Interface (Memory Access) [📁/忆 👓/见]

```
:maturity :stub
:depends-on [f0/P0, f1/P0]
```

+context:F1 is graph memory — fragments, relationships, annotations as nodes.

+if:The human needs to access, query, and navigate stored knowledge

+however:Currently F1 access requires direct API calls or Emacs functions

FUTON1

@multiarg futon1/devmap

@title FUTON1 Development Map — Deterministic Storage Substrate

@audience futon-devs, infra-agents, future-you

@tone formal-analytic

@style roadmap

@factor Keen investigation (dhammavicaya)

@IFR: FUTON1 provides a deterministic, storage-oriented substrate that maintains a coherent graph of facts across sessions. It keeps Datascript (fast) and XTDB (persistent) in reproducible alignment so that the same world state appears reliably across restarts and environments. It exposes a minimal CLI and HTTP surface for ingesting text, inspecting entities and relations, and emitting stable summaries—such as focus headers—that downstream code can depend on without embedding persistence or orchestration logic.

@state API settled with evolution protocols; invariant enforcement active; hydration settled; graph-memory and NLP interface operational; open-world ingest working; query workflows developing (cross-ref futon3a); pilot storage partly implemented (fulab), MMCA storage planned.

@next Stabilize invariant coverage; expand pilot storage for MMCA experiments; deepen futon3a query integration.

The Argument

FUTON1 provides a deterministic, storage-oriented substrate that maintains a coherent graph of facts across sessions. It exposes an HTTP API as the canonical interface—settled, with protocols for evolution—through which external tools query and update the persistent store. Invariants are enforced as real invariants: any violation is a ” stop the line” moment that prevents silent corruption. Graph-memory defines the core schema—entities, relations, salience metadata—and coordinates mirroring between Datascript (fast) and XTDB (persistent) so the same world state appears reliably across restarts. The NLP interface provides deterministic tokenization and tagging that all ingest paths depend on. XTDB hydration restores state on boot, with fallback to legacy log+snapshot when XTDB is disabled. Open-world ingest allows background knowledge to accumulate outside interactive sessions while sharing the same persistent store. Query workflows make the stored graph inspectable; futon3a develops complementary patterns for retrieval. Storage for pilots supports experimental workflows—fulab runs are partly implemented; MMCA experiment storage is planned.

! instantiated-by: Prototype 0 — HTTP API (Canonical Interface) [🌏/及 📤/及]

```
:maturity :settled
:depends-on []
:evidence-for-settled [API in daily use for
90+ days; evolution protocols documented]
```

+context:The HTTP API is the canonical interface to FUTON1.

+if:External tools need to query and update the persistent store

+however:Without a stable API, clients must embed internal assumptions or scrape CLI output

+then:Expose a minimal, documented HTTP surface for submitting utterances, inspecting state, and retrieving deterministic summaries (focus headers)

+because:A settled API lets higher futons and external tools depend on FUTON1 without coupling to internals

+evidence: evidence[apps/api module implements HTTP endpoints via command-service] evidence[/api/alpha/meta/model/registry exposes live model descriptors] evidence[Focus headers and entity summaries emitted through same service as CLI] evidence[XTDB-backed state consistent across CLI and API surfaces]

+ NEXT-STEPS:
next[Document evolution protocol for API versioning] next[Add /api/alpha/entities/latest?type=... for efficient recent-entity queries] next[Ensure clients omitting profile config resolve to documented default]

FUTON2

@multiarg futon2/devmap

@title FUTON2 Development Map — Active Inference Engine & Ant Demonstrators

@audience futon-devs, simulation-agents, future-you

@tone formal-analytic

@style roadmap

@factor Energy (viriya)

@IFR: FUTON2 provides a narratable Active Inference engine whose ants are first demonstrators for AIF agent species. The same engine powers futon3 fulab agents and may drive futon5 MetaCA simulations—making FUTON2 the shared AIF substrate that turns insight into kinetic practice across the stack.

@state Core AIF stack (observe→perceive→affect→policy→act) running end-to-end with analyzer hooks. Observation property tests strong (440+ iterations). Hunger/tau goldens recorded (6 scenarios). Policy goldens recorded (5 scenarios). Microtraces locked (4 scenarios). Gaps: world mechanics contract undocumented; external stepping API missing; futon1/futon3 bridges planned but not implemented.

@next Formalize world mechanics contract; expose external stepping API for episode export; design futon1 graph adapter; expand policy goldens to 10 scenarios.

The Argument

FUTON2 provides a narratable Active Inference engine with the ant war-game as its first demonstrator. The AIF stack baseline documents the full pipeline—observe, perceive, affect, policy, act—with locked parameters and golden microtraces so the system behaves reproducibly across seeds and refactors. The observation layer is hardened with property tests ensuring all sensory channels normalize to [0,1] across hundreds of random worlds. The predictive coding microcycle produces per-step traces of hunger, tau, and error evolution that regression tests lock against golden scenarios. Hunger and precision dynamics have six golden sequences covering normal operation through starvation, overstimulation, and cargo stress. The policy layer surfaces EFE-based action rankings through a deterministic harness with golden scenarios for forage, return, explore, and defend modes. The world mechanics provide the environment contract that all agent behaviour depends on. Pivot streams and analyzers enable introspection, feeding metrics to higher futons. An external stepping API will let futon3 and futon5 drive episodes programmatically. Bridges to futon1 graph memory and futon3 proofwork will let AIF agents read fact graphs and contribute to shared missions. Viriya metrics will connect simulation dynamics to social activation, making FUTON2 a practice accelerator rather than a curiosity.

! instantiated-by: Prototype 0 — AIF Stack Baseline [🌝/四 🗿/刊]

```
:maturity :active
:depends-on []
:evidence-for-active [Stack running end-
to-end for 90+ days; 4 golden microtraces
locked; parameters documented]
```

+context:FUTON2 runs a full Active Inference loop in the ant war-game.

+if:You want a stable story showing how observation, prediction errors, precision, and actions integrate per tick

+however:Without a documented baseline, modules can drift and break silently

+then:Capture top-level architecture (sensory keys through world updates), lock hunger and predictive micro-step parameters, and maintain golden AIF microtraces

+because:A documented baseline prevents accidental drift before optimising policy, analyzers, or mechanics

+evidence: evidence[doc/stack-baseline.md documents 5-stage pipeline with locked parameters] evidence[src/ants/aif/core.clj orchestrates full tick via aif-step()] evidence[test/resources/goldens/microtraces.edn locks 4 golden perception traces] evidence[doc/trace/single_tick.edn + SVG provide deterministic trace artifacts]

+ NEXT-STEPS:
next[Add doc/trace-semantics.md explaining micro-step semantics] next[Document hunger-

FUTON3

@multiarg futon3/devmap

@title FUTON3 Development Map — Flexiformal Proofwork & Pattern Checking

@audience futon-devs, sandbox-agents, future-you

@tone formal-analytic

@style roadmap

@factor Joy / Rapture (pīti)

@IFR: FUTON3 turns messy activity into organised knowledge by checking work against shared patterns and producing auditable records that other futons can use to learn, improve, and act.

@state MUSN transport runs reliably, pattern canon operational in futon1 storage, check DSL exercised end-to-end, trails capture tatami sessions, workday instrumentation bridges aob-chatgpt. Gaps: similarity field not wired into APIs; intent embedding designed but not implemented; joy metrics not yet flowing to futon0; proof hooks still bespoke scripts.

@next Wire similarity field into pattern search API; implement intent embedding pipeline; standardise proof→graph and proof→energy exports; land joy metrics as futon0 service.

The Argument

FUTON3 turns messy activity into organised knowledge by checking work against shared patterns and producing auditable records. The MUSN transport provides stable IO contracts for pattern-related checking with deterministic transcripts. The pattern canon stores patterns with metadata and devmap links in futon1 storage. The check DSL evaluates pattern applicability and logs every check as a proof-state record. Trails capture proof obligations and how sessions discharged them. Workday instrumentation bridges aob-chatgpt and fubar.el so user interactions flow into the check pipeline. The similarity field enables pattern suggestions via sigil-distance and GloVe embeddings. Intent embedding types each proof step as (fruit, orb) pairs for Kolmogorov-style constructive proofs. The pattern workbench drafts candidate flexiargs from accumulated proof states. Joy metrics track empowerment—time-to-proof, obligations cleared, commitments enabled—and are provided to futon0. Proof hooks export checks to futon1 (graph) and futon2 (viriya). The training ground packages tutorials so humans and agents can practice flexiformal checks. System self-description makes stack components queryable entities in the hypergraph.

! instantiated-by: Prototype 0 — MUSN Transport Baseline [➰/双 🎑/了]

```
:maturity :active
:depends-on []
:evidence-for-active [Transport running 90+
days; event logging operational; pattern
checks flow through router]
```

+context:FUTON3 rides the MUSN transport (WS+HTTP bus, SAFE/ADMIN REPL, NDJSON ingest) for pattern-related checking.

+if:You want an execution substrate where pattern obligations can be posed, evaluated, and recorded with deterministic transcripts

+however:Without frozen contracts, check jobs cannot rely on stable log formats

+then:Freeze the transport contract for pattern-apply, gap-report, and trail-capture; provide golden transcripts; version the sandbox profile as the canonical proofwork runtime

+because:Flexiformal checking needs a stable log format before any higher-layer reasoning can rely on it

+evidence: evidence[docs/protocol.md documents transport contracts] evidence[docs/protocol/transport-contract-v1.md freezes check-job contract] evidence[docs/protocol/golden-transcripts.md lists golden transcript fixtures] evidence[src/f2/router.clj implements event routing] evidence[test/f2/router_test.clj has 100-event idempotency + acceptance tests] evidence[test/f2/transport_golden_test.clj replays golden transcripts] evidence[test/fixtures/transport/golden-check-request.ndjson anchors deterministic transcripts] evidence[test/transport_test.clj covers per-client disconnect logging]

+ NEXT-STEPS:
next[Wire golden transcript replay into CI]

FUTON4

@multiarg futon4/devmap

@title FUTON4 Development Map — Memory Atelier & Hypertext Editing

@audience futon-devs, pattern-agents, future-you

@tone formal-analytic

@style roadmap

@factor Tranquility (passaddhi)

@IFR: FUTON4 is a memory workspace where documents, design patterns, and annotations share a common editing environment, letting symbolic artefacts be edited, linked, and narrated in one continuous workspace.

@state Arxana reanimation complete (storage bridge, browsing, relations, 80+ tests). Scholium mode has inclusion/transclusion previews with toggle commands. Pattern template support operational. Gaps: graph storage layer not wired to Datascript; XTDB integration pending; multi-user and literary interfaces planned.

@next Complete scholium UX for main article buffer; wire storage layer to Datascript; begin XTDB integration for version history.

The Argument

FUTON4 is a memory workspace where documents, design patterns, and annotations share a common editing environment. A key use case is to align software documentation and code, and for this reason an early prototype makes the system self-documenting. Arxana is a hypertext editor that links document fragments and lets you annotate them with scholia. Scholium mode shows how content is included or transcluded from other documents, with visual previews. Design patterns in a standard template are supported, translating into graph nodes for both human and machine editing. Futon4 stores text fragments and hyperlinks as addressable nodes so they can be queried and linked. XTDB adds version history so you can see how documents evolved over time. Multi-user support lets collaborators share a semantic space. A literary interface supports long-form writing like books and manuscripts.

! instantiated-by: Prototype 0 — Self-Documenting Foundations [🌽/久 🙅/才]

```
:maturity :settled
:depends-on []
:evidence-for-settled [Tangling machinery
operational; bootstrap harness tested;
manifesto documented]
```

+context:A key use case for futon4 is to align software documentation and code.

+if:You want documentation and implementation to stay synchronized

+however:Without self-documenting infrastructure, docs and code drift apart

+then:Make the system self-documenting with tangling machinery, bootstrap harness, and development manifesto

+because:This creates an anchor for all subsequent prototypes

+evidence: evidence[spine.org provides master orchestration with tangling machinery (lines 19-103)] evidence[dev/bootstrap.el is repeatable build harness for interactive + batch builds] evidence[AGENTS.md is development manifesto describing Phase 1/2 strategy] evidence[docs/reanimation-plan.org tracks M0-M6 milestones with checkbox progress] evidence[test/arxana-bootstrap-test.el validates baseline harness (3 tests)] evidence[arxana-tangled.el is generated single-source output (3752 lines) proving round-trip]

+ NEXT-STEPS:
next[Maintain synchronization as system evolves]

`:success-criteria`
pass[Tangling produces consistent output] pass[Bootstrap harness runs without errors] pass[Documentation reflects current implementation] fail[Docs and code diverge silently]

```
:psr-example ” Selected self-documenting
foundations because aligned docs reduce
maintenance burden”
:pur-template ” Tangle run, output:
{{file}}, lines: {{count}}, errors:
{{errors}}”
```

FUTON5

@multiarg futon5/devmap

@title FUTON5 Development Map — Design Patterns for Improvisation

@audience futon-devs, improviser-agents, future-you

@tone formal-analytic

@style roadmap

@factor Concentration (samādhi)

@IFR: FUTON5 builds and collects general-purpose design patterns for improvisation, with several demonstrator instances.

@state Nonstarter spec drafted with reference implementation in progress. Wiring diagrams operational with CT interpretation. MetaCA demonstrator running. Transfer from MetaCA to ants proven. Gaps: Nonstarter futon-stack deployment not yet running; dynamic programming for sequencing not implemented.

@next Complete Nonstarter futon-stack deployment; implement dynamic programming for action sequencing in music domain.

The Argument

FUTON5 builds and collects general-purpose design patterns for improvisation, with several demonstrator instances. Nonstarter is a crowdfunding specification where donations go to a shared pool and crowd voting determines allocation, with mechanisms to prevent popularity capture. One demo deployment of Nonstarter tracks the investment of energy into the development of the futon stack. Patterns in futon5 have a category-theoretic interpretation as wiring diagrams. MetaCA is a demonstrator domain where wiring diagrams control how cellular automata evolve. The transfer milestone proves wiring diagrams developed in one domain work in another—for example, from MetaCA to the ant simulation in futon2. Dynamic programming manages sequencing of actions in demonstration domains including music.

! instantiated-by: Prototype 0 — Nonstarter Specification [🎶/己 🔥/世]

```
:maturity :active
:depends-on []
:evidence-for-active [Spec drafted;
reference implementation in progress]
```

+context:Nonstarter is a crowdfunding specification for pool-based allocation.

+if:You want crowdfunding where the crowd decides allocation rather than donors picking projects

+however:Traditional crowdfunding ties donations to specific projects

+then:Pool donations, let crowd vote on allocation, auto-fund when threshold met, decay votes to prevent popularity capture

+because:Pool-based allocation with vote decay resists capture and funds unlikely ideas

+evidence: evidence[docs/nonstarter_spec.md defines pool funding, voting, auto-funding, governance] evidence[docs/nonstarter-adapter-contract.md defines adapter interface] evidence[docs/nonstarter-terminal-vocabulary.md defines terminal concepts] evidence[scripts/nonstarter_propose.clj implements proposal logic]

+ NEXT-STEPS:
next[Complete reference implementation] next[Test voting and auto-funding mechanics] next[Document governance primitives]

`:success-criteria`
pass[Pool funding operational] pass[Voting with decay implemented] pass[Auto-funding triggers correctly] fail[Popularity capture occurs]

```
:psr-example ” Selected Nonstarter for
futon-stack funding because pool allocation
resists capture”
:pur-template ” Nonstarter event {{type}},
pool: {{balance}}, proposals: {{count}}”
```

! instantiated-by: Prototype 1 — Nonstarter Futon-Stack Deployment [🌲/习 🔥/己]

```
:maturity :greenfield
:depends-on [f5/P0]
:evidence-for-active [Deployment tracking
energy investment into futon stack]
```

+context:One demo deployment of Nonstarter

FUTON6

@multiarg futon6/devmap

@title FUTON6 Development Map — Comprehensive Mathematics Dictionary

@audience futon-devs, pattern-agents, HDM-stewards, future-you

@tone formal-analytic

@style roadmap

@factor Equanimity (upekkhā)

@IFR: FUTON6 is a comprehensive mathematics dictionary where informal and formal arguments coexist, indexed by patterns at multiple levels of abstraction.

@state Design-only vision; no dedicated code or imports exist yet beyond conceptual outlines.

@next Define seed domain; implement first concrete entries; design reasoning map.

The Argument

FUTON6 is a comprehensive mathematics dictionary. Informal arguments from F3 are supported alongside formal proofs—F3 manages everyday reasoning patterns like ” look for an easier problem” or ” fully work a set of relevant examples” . A seed domain provides the first concrete entries: definitions, examples, small proofs. Cross-domain reasoning patterns name moves like quotient constructions that work across many fields. Domain-specific patterns name constructs like ultrafilters that are particular to one area. A reasoning map shows how informal, cross-domain, and domain-specific patterns fit together to make mathematics work. Interactive tutorials let people practice by working through problems. Landscape mode shows how entries connect to neighbours. StackExchange import seeds the dictionary with Q&A pairs. An agent protocol defines what queries and edits are permitted. General purpose mathematical models are used to describe the growth and development of futon6 and are presented in a way that will make them reusable in other knowledge domains.

! instantiated-by: Prototype 0 — Informal Argument Support [👎/已 🍍/双]

```
:maturity :greenfield
:depends-on [f3/P1, f3/P2]
:evidence-for-active [Informal arguments
import from F3; metadata attached]
```

+context:F3 manages everyday reasoning patterns like ” look for an easier problem” .

+if:Mathematics should include informal reasoning alongside formal proofs

+however:Traditional dictionaries separate informal and formal content

+then:Import checked arguments from F3 with metadata; support both informal and formal entries

+because:Real mathematical practice uses informal reasoning patterns

+evidence: evidence[F3 pattern canon exists] evidence[F3 check DSL operational]

+ NEXT-STEPS:
next[Define import contract from F3] next[Attach math metadata to imported arguments] next[Test informal/formal coexistence]

`:success-criteria`
pass[Informal arguments import correctly] pass[Metadata preserved] pass[Both types navigable together] fail[Informal content rejected or lost]

```
:psr-example ” Selected informal argument
support because real mathematics uses
everyday reasoning”
:pur-template ” Import {{argument}},
type: {{informal|formal}}, metadata:
{{attached}}”
```

! instantiated-by: Prototype 1 — Seed Domain [🔸/小 💬/才]

```
:maturity :greenfield
:depends-on []
:evidence-for-active [Seed domain defined;
first entries created]
```

+context:A concrete seed domain proves viability.

+if:You want to demonstrate the dictionary works

+however:Abstract designs need concrete examples

+then:Choose a coherent seed domain and create linked entries: definitions, examples,

FUTON7

@multiarg futon7/devmap

@title FUTON7 Development Map — Transparent Knowledge Economy

@audience futon-devs, sphere-architects, future-you

@tone formal-analytic

@style roadmap

@factor Civic equanimity (upa-upekkhā)

@IFR: FUTON7 is a new form of knowledge economy that adds transparency to the regular economy by tracking costs, relationships, and externalities that current systems obscure.

@state Design-only narratives; Hyperreal operates as informal seed institution; Gravpad conceptualised but not implemented; scenario work not yet structured.

@next Define Gravpad MVP; structure first scenario simulation; document Hyperreal as explicit F7 prototype.

The Argument

FUTON7 is a new form of knowledge economy that adds transparency to the regular economy by tracking costs, relationships, and externalities that current systems obscure. Hyperreal Enterprises is a seed institution testing these ideas in real economic terrain. Gravpad does to the web what street art does to cities—adding commentary and transparency without permission. Mathematical models from F6 feed into scenario simulations that test future structures before implementation.

! instantiated-by: Prototype 0 — Hyperreal Enterprises as Seed Institution [⌛/己 💫/业]

```
:maturity :greenfield
:depends-on []
:evidence-for-active [Hyperreal operating;
governance documented; economic model
tested]
```

+context:F7 needs a living prototype, not just theory.

+if:You want to test transparent knowledge economy ideas in real conditions

+however:Abstract designs need concrete instantiation

+then:Position Hyperreal explicitly as F7’ s seed institution practicing transparent governance, pattern-based documentation, and consulting as economic model

+because:A concrete institution anchors abstractions into lived practice

+evidence: evidence[Hyperreal operates as consulting practice] evidence[Values and narrative documented] evidence[Client-based R&D loops exist]

+ NEXT-STEPS:
next[Document Hyperreal governance explicitly as F7 prototype] next[Track economic flows transparently] next[Iterate institutional design based on experience]

`:success-criteria`
pass[Hyperreal operates sustainably] pass[Governance transparent and documented] pass[Learnings feed back into F7 design] fail[Institution collapses or becomes opaque]

```
:psr-example ” Selected Hyperreal as seed
because living prototypes test theory”
:pur-template ” Hyperreal period {{period}},
clients: {{count}}, governance updates:
{{updates}}”
```

! instantiated-by: Prototype 1 — Gravpad [👎/引 ⛲/广]

```
:maturity :greenfield
:depends-on []
:evidence-for-active [Gravpad deployed; web
annotation operational]
```

+context:Gravpad does to the web what street art does to cities.

+if:You want to add commentary and transparency to the web without permission

+however:Current web infrastructure serves platform owners, not users

+then:Build Gravpad as web annotation layer that adds traceability, commentary, and accountability information to existing web content

+because:Transparency infrastructure should not require permission from those being made

+then:Surface memory access through F0 — query from HUD, see relationships, navigate links
+because:If the human cannot easily access memory, F1 does not extend their capability

+ NEXT-STEPS:
next[Add memory query widget to Stack HUD] next[Surface recent fragments and their relationships] next[Enable link navigation from HUD context]

:success-criteria
pass[Human can query F1 without leaving the HUD context] pass[Relationships are visible and navigable] fail[F1 access remains a separate, friction-full operation]

! instantiated-by: Prototype 2 — F0-F2 Interface (Agent Perception) [📎/才 👀/见]
:maturity :stub
:depends-on [f0/P0, f2/P0]
+context:F2 is active inference — perception-action loops for agents.
+if:The human needs to perceive what agents are doing and direct their actions
+however:Currently agent sessions are logged but not surfaced in real-time
+then:Surface agent state through F0 — see current beliefs, policies, actions; redirect when needed
+because:If the human cannot perceive agents, they cannot effectively collaborate with them

+ NEXT-STEPS:
next[Surface active agent sessions in Stack HUD] next[Show agent beliefs and current policy] next[Enable human redirection of agent focus]

:success-criteria
pass[Human can see what agents are doing without reading logs] pass[Human can redirect agents through F0 interface] fail[Agent state remains opaque until session ends]

! instantiated-by: Prototype 3 — F0-F3 Interface (Verification Visibility) [😄/正 📁/正]
:maturity :stub
:depends-on [f0/P0, f3/P0]
+context:F3 is pattern checking — validation of informal arguments.
+if:The human needs to see verification status of claims and arguments
+however:Currently checks are logged but not surfaced to F0
+then:Surface verification status through F0 — see which claims are checked, what evidence exists, what is missing
+because:If the human cannot see verification status, they cannot trust or improve arguments

+ NEXT-STEPS:
next[Surface recent checks in Stack HUD] next[Show evidence status for active claims] next[Highlight missing evidence or blocked checks]

:success-criteria
pass[Human can see verification status at a glance] pass[Missing evidence is visible and actionable] fail[Verification remains invisible until explicitly queried]

! instantiated-by: Prototype 4 — F0-F4 Interface (Hypertext Navigation) [📁/文 ➰/久]
:maturity :stub
:depends-on [f0/P0, f4/P0]
+context:F4 is hypertext editing — documents as graph nodes with relationships.
+if:The human needs to navigate documents and their relationships
+however:Currently document navigation is through Emacs/Arxana directly
+then:Surface document structure through F0 — see current document context, related documents, annotations
+because:If the human cannot navigate hypertext fluidly, document relationships remain implicit

+ NEXT-STEPS:
next[Surface document context in Stack HUD] next[Show related documents and annotations] next[Enable quick navigation to

:success-criteria
pass[API responses remain stable across tagged versions] pass[External clients can query and update graph without internal knowledge] pass[/status surface shows resolved datastore path and config] fail[Breaking changes without version bump or migration path]

:psr-example ” Selected HTTP API for futon3a integration because it’ s the canonical FUTON1 interface”
:pur-template ” Used API endpoint {{endpoint}}, response time: {{ms}}ms, status: {{success|error}}”

! instantiated-by: Prototype 1 — XTDB Hydration [🌊/水 🔃/反]
:maturity :settled
:depends-on [f1/P3]
:evidence-for-settled [Hydration verified across 100+ restart cycles; fallback to legacy tested]
+context:XTDB hydration restores world state on boot.
+if:Sessions must resume with the same state they ended with
+however:Without reliable hydration, restarts lose or corrupt state
+then:Hydrate from XTDB on boot, with fallback to legacy log+snapshot when XTDB is disabled
+because:Settled hydration makes FUTON1 a durable substrate, not just a session-scoped cache
+evidence: evidence[Sessions hydrate from XT on boot] evidence[Fallback to legacy log+snapshot when XTDB disabled] evidence[Salience-survives-restart demonstration is canonical hydration test] evidence[-M:storage/inspect demonstrates reproducible hydration]

+ NEXT-STEPS:
next[Document hydration sequence in storage reference] next[Add hydration timing telemetry]

:success-criteria
pass[State after hydration matches state before shutdown] pass[Fallback engages correctly when XTDB unavailable] pass[Hydration completes within reasonable time bounds] fail[State loss or corruption across restart]

:psr-example ” Selected XTDB hydration for session continuity because durability requires reliable restore”
:pur-template ” Hydration from {{source}}, entities loaded: {{count}}, duration: {{ms}}ms”

! instantiated-by: Prototype 2 — Invariant Enforcement [👉/止 😄/正]
:maturity :active
:depends-on [f1/P0]
:evidence-for-settled [Zero silent corruptions over 60+ days; all invariant violations logged and halted]
+context:Invariants must be real invariants, not suggestions.
+if:The storage substrate promises determinism and consistency
+however:Without enforcement, violations can corrupt state silently
+then:Treat invariant violations as ” stop the line” moments—halt processing, log the violation, prevent silent corruption
+because:A deterministic substrate is only trustworthy if invariants are actually enforced
+evidence: evidence[Open-world invariants pass end-to-end including stub detection and relation type registration] evidence[CI green on Futon1 Deterministic Stack tests] evidence[Invariant failures halt processing rather than continuing with corrupt state]

+ NEXT-STEPS:
next[Audit remaining invariant coverage gaps] next[Add invariant violation telemetry for monitoring] next[Document invariant categories and their enforcement points]

:success-criteria
pass[Any invariant violation halts processing immediately] pass[Violations are logged with sufficient context for diagnosis] pass[No silent corruption reaches persistent store] fail[Invariant

tau coupling (when update-tau fires vs perceive-internal updates)] next[Add 4 complete tick traces with commentary (forage, return, explore, defend)]

:success-criteria
pass[Golden microtraces pass regression tests] pass[Parameters locked in stack-baseline.md] pass[New modules can integrate without breaking existing traces] fail[Microtrace regression without version bump or migration]

:psr-example ” Selected AIF stack baseline for fulab integration because it’ s the reference AIF implementation”
:pur-template ” AIF tick {{tick-id}}, perception steps: {{steps}}, final tau: {{tau}}”

! instantiated-by: Prototype 1 — Observation Layer Hardening [🌝/四 👎/刊]
:maturity :active
:depends-on [f2/P0]
:evidence-for-active [440+ property test iterations; 14-channel vector ABI locked]
+context:g-observe normalises sensory evidence into vectors for downstream processing.
+if:You want deterministic observations across seeds and a nailed-down vector ABI
+however:Without property tests, normalisation bounds and edge cases can break silently
+then:Enforce 0–1 ranges via property tests, document sense→vector ordering, and test edge cases (borders, degenerate fields)
+because:Reliable observations keep perception and policy stable across world configurations
+evidence: evidence[test/ants/aif/observe_test.clj runs 200+ random worlds with g-observe-values-clamped] evidence[120 iterations test border/corner neighbor calculations] evidence[Degenerate field test handles max-food=0 gracefully] evidence[14-element vector ABI locked in sense-vector-ordering test]

+ NEXT-STEPS:
next[Add doc/observation-spec.md listing all normalization bounds] next[Add white-space detection property tests (:white? field currently untested)] next[Update README.md observation vector docs (lists 13, actual is 14)]

:success-criteria
pass[All sensory channels return values in [0,1]] pass[Vector ABI stable across refactors] pass[Edge cases (borders, empty fields) handled correctly] fail[Observation varies non-deterministically for same input]

:psr-example ” Selected observation hardening for ML export because deterministic vectors are prerequisite”
:pur-template ” Observation test run, iterations: {{count}}, channels: 14, all clamped: {{yes|no}}”

! instantiated-by: Prototype 2 — Predictive Coding Microcycle [🙇/乡 📤/叉]
:maturity :active
:depends-on [f2/P0]
:evidence-for-active [4 golden scenarios locked; regression tests pass; trace tooling operational]
+context:Perception performs multi-step updates of mu, tau, Pi-o, goals, and weighted errors.
+if:Inspectors or future bridges need to understand belief updates
+however:Without canonical traces, hunger updates and error accumulation are opaque
+then:Publish single-tick perception traces with per-step hunger, tau, and error; lock against golden scenarios
+because:Clear cycles make integrations with futon4 symbolic memory and futon3 fulab far easier
+evidence: evidence[test/resources/goldens/microtraces.edn has 4 golden scenarios with per-step traces] evidence[test/ants/aif/microtrace_regression_test.clj compares live runs to goldens] evidence[tools/trace_tick.clj generates doc/trace/single_tick.edn + SVG] evidence[src/ants/aif/perceive.clj outputs {tau, h, error} per micro-step]

+ NEXT-STEPS:
next[Add doc/trace-semantics.md formalizing

next[Document sandbox profile as canonical proofwork runtime]

:success-criteria
pass[Transport contract frozen and versioned] pass[Golden transcripts reproducible from sandbox profile] pass[Check jobs have stable log formats] fail[Contract changes break downstream consumers]

:psr-example ” Selected MUSN transport for check jobs because stable IO contracts enable reproducible proofwork”
:pur-template ” Transport event {{type}}, client: {{client}}, logged: {{yes|no}}”

! instantiated-by: Prototype 1 — Pattern Canon & Standard Library [🐺/本 ➰/世]
:maturity :active
:depends-on [f1/P0]
:evidence-for-active [Pattern store operational in futon1; ingest working; tests pass]
+context:A flexiarg/pattern library exists but was scattered across files.
+if:You want FUTON3 to curate patterns with metadata, linking obligations, and proof status
+however:Without a canonical store, the checker has nothing trustworthy to compare against
+then:Stand up f3.patterns store (ID, clauses, references, pāramitā tags, fruits/orbs weights) in futon1 storage; keep in sync with filesystem devmaps as ” checked out working copy”
+because:Pattern-based inference requires stable, addressable patterns that can be inspected and edited across the stack
+evidence: evidence[Pattern store working in futon1 with ingest] evidence[resources/pattern_store.edn maintains catalog] evidence[src/futon3/pattern_store.clj implements store API] evidence[test/pattern_store_test.clj verifies sync with README references]

+ NEXT-STEPS:
next[Add golden test proving store loads and enumerates all pattern IDs] next[Add golden test asserting metadata completeness (ID, clauses, refs, tags, weights)] next[Verify pattern store matches standard library with no omissions]

:success-criteria
pass[Pattern store loads at runtime with stable set of IDs] pass[Each pattern has required metadata and valid source links] pass[Store matches filesystem library exactly] fail[Patterns only exist on disk, not accessible to stack]

:psr-example ” Selected pattern canon for check DSL because addressable patterns enable programmatic checking”
:pur-template ” Pattern {{id}} loaded, clauses: {{count}}, tags: {{tags}}”

! instantiated-by: Prototype 2 — Check DSL & Applicability Engine [🌽/未 ➰/广]
:maturity :active
:depends-on [f3/P0, f3/P1]
:evidence-for-active [check! API operational; schema validation working; checks logged]
+context:FUTON3 answers ” can this pattern apply?” and ” what obligations arise?”
+if:You want a declarative DSL for posing check contexts and receiving structured verdicts
+however:Without executable checks, patterns remain narrative rather than flexiformal
+then:Define check! API (inputs: pattern ID, context EDN, evidence refs; outputs: status, missing fields, derived tasks); enforce schema validation; log every check as proof-state record
+because:Turning patterns into executable checks is what makes FUTON3 flexiformal
+evidence: evidence[src/futon3/checks.clj implements check DSL] evidence[src/f2/router.clj routes check requests] evidence[src/futon3/futon1_bridge.clj wires to futon1] evidence[src/futon3/futon2_bridge.clj logs viriya deltas for checks] evidence[src/f2/transport.clj exposes /musn/check HTTP endpoint] evidence[docs/api/musn.yaml documents /musn/check] evidence[test/transport_test.clj covers workday/check forwarding]

+ NEXT-STEPS:
next[Add check latency instrumentation]

! instantiated-by: Prototype 1 — Arxana Hypertext Editor [🙍/久 💖/用]
:maturity :settled
:depends-on [f4/P0]
:evidence-for-settled [Spine parser rebooted; node browsing restored; storage bridge complete; 80+ tests pass]
+context:Arxana is a hypertext editor that links document fragments and annotations.
+if:You want stable parsing, node navigation, and save-back working
+however:Legacy structures were dormant and needed reanimation
+then:Reboot spine parser, restore node browsing, define nema-based schema, implement storage bridge, test round-trip edits
+because:A functioning editor is the precondition for all downstream memory work
+evidence: evidence[spine.org lines 50-103 define arxana-tangle-spine-concat] evidence[dev/arxana-relations.el provides ego, cooccur, tail commands] evidence[dev/arxana-browse.el provides catalog, labels, history navigation] evidence[0-Reintroduction.org documents EAV model (nema, article, metadata scholium, event/hyperedge, plexus)] evidence[15 test files in test/ with ~80 tests total] evidence[dev/arxana-store.el (350+ lines) with 12 Futon API helpers] evidence[docs/storage-bridge.org documents API with curl examples] evidence[dev/arxana-article.el handles metadata sync and deletion propagation] evidence[dev/arxana-scholium.el provides compose, region capture, authoring mode]

+ NEXT-STEPS:
next[Produce migration notes for historical datasets (M5 item)] next[Add CI configuration for automated testing] next[Tag ” revived” release with reproducible steps]

:success-criteria
pass[Spine parser produces correct output] pass[Node browsing navigates document graph] pass[Storage bridge syncs with futon1] pass[Round-trip edits preserve content] fail[Edits corrupt document structure]

:psr-example ” Selected Arxana reboot for hypertext editing because linked documents need stable infrastructure”
:pur-template ” Arxana session, nodes visited: {{count}}, edits: {{edits}}, saved: {{yes|no}}”

! instantiated-by: Prototype 2 — Scholium Mode [🐲/久 🙍/用]
:maturity :active
:depends-on [f4/P1]
:evidence-for-active [Derivation previews working; toggle commands implemented; 5 tests pass]
+context:Scholium mode shows how content is included or transcluded from other documents.
+if:You want to see document relationships with visual affordances
+however:Inclusion/transclusion relationships were not visually differentiated
+then:Add collapsible previews, color overlays, and toggle commands for inclusion/transclusion/identification highlights
+because:Visual affordances make document structure navigable
+evidence: evidence[dev/arxana-derivation.el (290+ lines) provides preview blocks and face definitions] evidence[arxana-derivation-toggle-inclusion-highlights command implemented] evidence[arxana-derivation-toggle-transclusion-highlights command implemented] evidence[arxana-derivation-toggle-identification-highlights command implemented] evidence[test/arxana-derivation-test.el covers preview detection, rendering, toggle (5 tests)]

+ NEXT-STEPS:
next[Wire main article buffer overlays to mark-things-up hook] next[Document visual affordances in docs/inclusion-ui.md] next[Add integration test for main buffer + display buffer side-by-side]

:success-criteria
pass[Previews render correctly in display buffer]

tracks energy investment into the futon stack.
+if:You want to track how effort is allocated across futon development
+however:No current mechanism tracks energy investment with Nonstarter mechanics
+then:Deploy Nonstarter instance for futon stack; track contributions, votes, and funded work
+because:Eating your own cooking validates the specification
+evidence: evidence[Nonstarter spec available] evidence[Futon stack has trackable work items]

+ NEXT-STEPS:
next[Set up Nonstarter instance for futon stack] next[Define proposal categories for futon work] next[Track energy investment over time]

:success-criteria
pass[Nonstarter instance running for futon stack] pass[Energy investment tracked] pass[Allocation decisions made via voting] fail[Deployment not operational]

:psr-example ” Selected futon-stack deployment for Nonstarter validation because self-hosting proves viability”
:pur-template ” Futon energy {{period}}, invested: {{hours}}, allocated: {{distribution}}”

! instantiated-by: Prototype 2 — Wiring Diagrams as Patterns [🌲/归 🔃/今]
:maturity :active
:depends-on []
:evidence-for-active [CT interpretation documented; wiring diagrams operational in MetaCA]
+context:Patterns in futon5 have a category-theoretic interpretation as wiring diagrams.
+if:You want patterns that compose and transfer between domains
+however:Informal patterns lack compositional semantics
+then:Interpret patterns as wiring diagrams with CT semantics; components are morphisms, composition is diagram gluing
+because:Wiring diagrams give patterns precise compositional meaning that transfers across domains
+evidence: evidence[Wiring diagrams implemented in src/futon5/wiring/] evidence[Hexagram-to-wiring mapping in src/futon5/wiring/hexagram.clj] evidence[Runtime executes wiring diagrams as CA rules] evidence[256 exotypes documented in futon3/library/iiching/]

+ NEXT-STEPS:
next[Document CT interpretation formally] next[Add more wiring diagram primitives] next[Test composition laws]

:success-criteria
pass[Wiring diagrams compose correctly] pass[CT interpretation consistent] pass[Diagrams transfer between domains] fail[Composition breaks semantics]

:psr-example ” Selected wiring diagrams for pattern representation because CT gives compositional semantics”
:pur-template ” Wiring diagram {{id}}, components: {{count}}, composed: {{yes|no}}”

! instantiated-by: Prototype 3 — MetaCA Demonstrator [🔃/己 🔺/今]
:maturity :active
:depends-on [f5/P2]
:evidence-for-active [MetaCA running; wiring diagrams control CA evolution; test sheet generated]
+context:MetaCA is a demonstrator domain where wiring diagrams control how cellular automata evolve.
+if:You want a computational substrate for testing pattern evolution
+however:Abstract patterns need concrete execution environments
+then:Use MetaCA as demonstrator where wiring diagrams define local evolution rules; global constraints from AIF
+because:MetaCA provides executable semantics for wiring diagram patterns
+evidence: evidence[src/futon5/ca/core.clj

small proofs, scholia
+because:A tangible neighbourhood demonstrates the approach
+evidence: evidence[Seed domain candidates identified (compactness, categorical arrows, Abelian groups)]

+ NEXT-STEPS:
next[Select seed domain] next[Create first 10-20 entries] next[Link entries with cross-references]

:success-criteria
pass[Seed domain entries exist] pass[Entries linked coherently] pass[Examples and proofs included] fail[Entries isolated or incomplete]

:psr-example ” Selected seed domain for proof of concept because concrete examples validate design”
:pur-template ” Seed entry {{id}}, type: {{def|example|proof}}, links: {{count}}”

! instantiated-by: Prototype 2 — Cross-Domain Reasoning Patterns [💬/乃 🔺/今]
:maturity :greenfield
:depends-on [f6/P1]
:evidence-for-active [Cross-domain patterns catalogued; entries tagged]
+context:Some reasoning moves work across many fields.
+if:You want to name and reuse general mathematical strategies
+however:These moves are often left implicit
+then:Catalogue cross-domain patterns like quotient constructions, local-to-global arguments, dualisation; tag entries with applicable patterns
+because:Named patterns become teachable and reusable
+evidence: evidence[Pattern candidates identified from mathematical practice]

+ NEXT-STEPS:
next[Catalogue 10-15 cross-domain patterns] next[Tag seed domain entries] next[Document pattern usage]

:success-criteria
pass[Patterns catalogued with examples] pass[Entries tagged appropriately] pass[Patterns reused across domains] fail[Patterns remain implicit]

:psr-example ” Selected cross-domain patterns for reuse because general moves should be named”
:pur-template ” Pattern {{id}}, domains: {{count}}, uses: {{entries}}”

! instantiated-by: Prototype 3 — Domain-Specific Patterns [🔺/今 👉/乃]
:maturity :greenfield
:depends-on [f6/P1]
:evidence-for-active [Domain-specific patterns catalogued; entries tagged]
+context:Each mathematical area has characteristic constructs.
+if:You want to capture domain-specific machinery
+however:These constructs are often described only within their home domain
+then:Catalogue domain-specific patterns like ultrafilters, Yoneda, compact operators, Markov kernels; document canonical usages
+because:Named constructs make each domain navigable
+evidence: evidence[Domain constructs identified from seed domain]

+ NEXT-STEPS:
next[Catalogue 10-20 domain-specific patterns for seed domain] next[Document canonical usages] next[Link to cross-domain patterns where applicable]

:success-criteria
pass[Domain patterns catalogued] pass[Canonical usages documented] pass[Links to cross-domain patterns] fail[Domain machinery undocumented]

:psr-example ” Selected domain-specific patterns for navigability because each field has characteristic tools”
:pur-template ” Domain pattern {{id}},

transparent
+evidence: evidence[Arxana provides annotation model] evidence[Web annotation standards exist]

+ NEXT-STEPS:
next[Define Gravpad MVP] next[Build annotation layer] next[Test on real web content] next[Document supply chain / externality use cases]

:success-criteria
pass[Annotations attach to web content] pass[No permission required from target sites] pass[Traceability information visible to users] fail[Annotations blocked or invisible]

:psr-example ” Selected Gravpad for transparency because street art doesn’ t ask permission”
:pur-template ” Gravpad annotation {{url}}, type: {{type}}, visible: {{yes|no}}”

! instantiated-by: Prototype 2 — Scenario Simulations [🚃/习 🔸/扎]
:maturity :greenfield
:depends-on [f6/P9]
:evidence-for-active [Scenario bundles defined; simulations run; results documented]
+context:Future structures should be tested before implementation.
+if:You want to prototype civic and economic forms before committing
+however:Abstract philosophy does not test well
+then:Develop scenario bundles tied to F6 mathematical models; run simulations; document results and learnings
+because:Scenario practice turns philosophy into actionable design
+evidence: evidence[F6 provides mathematical models] evidence[Scenario concepts identified]

+ NEXT-STEPS:
next[Define 2-3 scenario bundles] next[Tie scenarios to F6 models] next[Run simulations] next[Document learnings]

:success-criteria
pass[Scenarios defined with clear parameters] pass[Simulations produce testable results] pass[Learnings inform F7 design] fail[Scenarios remain abstract]

:psr-example ” Selected scenario simulations for testing because future structures need prototyping”
:pur-template ” Scenario {{name}}, model: {{model}}, result: {{outcome}}”

linked fragments]

:success-criteria
pass[Human can see document relationships from HUD] pass[Navigation between related documents is fluid] fail[Document relationships require manual search]

! instantiated-by: Prototype 5 — F0-F5 Interface (Pattern Work) [💎/本 🔃/去]
:maturity :stub
:depends-on [f0/P0, f5/P0]
+context:F5 is wiring diagrams — compositional pattern transfer across domains.
+if:The human needs to work with patterns, see transfers, compose diagrams
+however:Currently pattern work requires understanding category theory notation
+then:Surface pattern structure through F0 — see active patterns, visualize wiring, understand transfers
+because:If the human cannot perceive patterns, they cannot leverage compositional abstraction

+ NEXT-STEPS:
next[Surface active patterns and their domains in Stack HUD] next[Visualize wiring diagrams in accessible form] next[Show pattern transfers between domains]

:success-criteria
pass[Human can see which patterns are active and where] pass[Wiring diagrams are comprehensible without CT background] fail[Pattern work remains expert-only]

! instantiated-by: Prototype 6 — F0-F6 Interface (Mathematical Access) [✌️/计 📁/书]
:maturity :stub
:depends-on [f0/P0, f6/P0]
+context:F6 is mathematical dictionary — definitions, proofs, examples with cross-references.
+if:The human needs random access to mathematical knowledge
+however:Currently math access is through linear reading or search
+then:Surface mathematical structure through F0 — query definitions, see dependencies, navigate proofs
+because:If the human cannot access math nonlinearly, the dictionary structure provides no advantage

+ NEXT-STEPS:
next[Add math query capability to Stack HUD] next[Surface definition dependencies and examples] next[Enable proof navigation from HUD context]

:success-criteria
pass[Human can query mathematical concepts from HUD] pass[Dependencies between definitions are visible] fail[Math access remains search-based with no structure]

! instantiated-by: Prototype 7 — F0-F7 Interface (Transparency/Simulation) [👓/入 💖/本]
:maturity :stub
:depends-on [f0/P0, f7/P0]
+context:F7 is transparency and civic simulation — annotations on web content, scenario modeling.
+if:The human needs to see transparency annotations and run simulations
+however:Currently these are separate browser extensions and notebooks
+then:Surface transparency and simulation through F0 — see annotations, run scenarios, understand models
+because:If the human cannot access transparency tools fluidly, economic structures remain opaque

+ NEXT-STEPS:
next[Surface active annotations in Stack HUD] next[Enable scenario simulation from HUD context] next[Show model assumptions and results]

:success-criteria
pass[Human can see transparency annotations

violation proceeds without halt or log]

:psr-example ” Selected invariant-enforcement for storage integrity because silent corruption is unacceptable”
:pur-template ” Invariant check {{invariant-id}}, result: {{pass|halt}}, context: {{details}}”

! instantiated-by: Prototype 3 — Graph-Memory Schema & Mirroring [🍵/文 🔃/反]
:maturity :active
:depends-on []
:evidence-for-settled [Schema stable for 90+ days; mirroring verified across restart cycles]
+context:Graph-memory is the core data model for FUTON1.
+if:FUTON1 must maintain a coherent fact graph across sessions
+however:Without a clear schema and mirroring contract, clients cannot rely on stored state
+then:Define entity/relation schema, salience metadata, and Datascript↔XTDB mirroring rules as a documented contract
+because:A clear graph-memory contract turns internal storage into a reliable interface
+evidence: evidence[apps/graph-memory defines Datascript schema and XTDB touchpoints] evidence[Test suites exercise persistence and salience fields] evidence[XT inspection paths demonstrate entity/relation document storage and reload] evidence[README documents schema, hydration behaviour, and mirroring]

+ NEXT-STEPS:
next[Extract core graph-memory operations as named public API] next[Document which document types and fields are mirrored into XTDB] next[Describe expected seed graph in module README] next[Add migration notes for any schema changes]

:success-criteria
pass[Schema changes follow documented migration protocol] pass[Mirroring produces identical state in Datascript and XTDB] pass[External tools can query graph using documented shapes] fail[Schema drift between Datascript and XTDB representations]

:psr-example ” Selected graph-memory for entity storage because it defines FUTON1’ s core data model”
:pur-template ” Graph operation {{op}}, entities affected: {{count}}, mirrored: {{yes|no}}”

! instantiated-by: Prototype 4 — NLP Interface [📁/义 🎴/印]
:maturity :active
:depends-on []
:evidence-for-settled [Deterministic outputs verified across 50+ test fixtures]
+context:The NLP interface provides deterministic text processing.
+if:All ingest paths need consistent tokenization and tagging
+however:Without a clear pipeline, NLP behaviour is implicit in code
+then:Expose tokenise → tag → chunk → recognise as documented stages with testable outputs
+because:Deterministic NLP is the foundation for reproducible graph construction
+evidence: evidence[apps/nlp-interface provides deterministic NER/POS tagging] evidence[Pattern-recognition hooks used across ingest and demo/client workflows] evidence[Test runners exist for NLP fixtures] evidence[README describes deterministic behaviour and shared pipeline usage]

+ NEXT-STEPS:
next[Document pipeline stages as public API] next[Expand tests to cover each stage independently] next[Clarify expected input/output shapes for each stage]

:success-criteria
pass[Same input produces identical NLP output across runs and environments] pass[Pipeline stages are independently testable] pass[Upstream components agree on API usage] fail[NLP

micro-step semantics] next[Document goal-bias mechanism (blending toward home/enemy based on cargo)] next[Explain why errors are 0.0 in certain goldens]

:success-criteria
pass[Golden scenarios pass regression tests] pass[Traces show tau, h, error evolution per step] pass[External tools can extract mu/precision snapshots] fail[Perception output varies for same input and config]

:psr-example ” Selected predictive coding traces for fulab agent debugging because step-by-step visibility is essential”
:pur-template ” Microtrace scenario {{name}}, steps: {{count}}, final tau: {{tau}}, final h: {{h}}”

! instantiated-by: Prototype 3 — Hunger & Precision Dynamics [💫/六 🙇/叉]
:maturity :active
:depends-on [f2/P0]
:evidence-for-active [6 golden sequences locked; regression tests pass]
+context:tick-hunger, update-hunger, update-tau, and modulate-precisions control the most sensitive dynamics.
+if:Affective behaviour must survive refactors reproducibly
+however:Without goldens, tau evolution and hunger saturation can drift
+then:Record golden sequences for key scenarios (normal, starvation, overstimulation, heavy-cargo, high-risk, high-ingest) and lock expected tau floors/caps
+because:Once goldens exist, small tune-ups stop breaking entire colony behaviour
+evidence: evidence[test/resources/goldens/hunger_tau.edn has 6 golden sequences as specified] evidence[test/ants/aif/hunger_regression_test.clj runs all 6 scenarios through full affect pipeline] evidence[tools/hunger_goldens.clj generates golden file deterministically] evidence[README.md documents hunger config parameters and defaults]

+ NEXT-STEPS:
next[Add doc/hunger-goldens-guide.md explaining behavioral regime each scenario represents] next[Generate SVG plots for hunger_tau trajectories] next[Add tau-floor/tau-cap assertions to regression test] next[Add boundary saturation tests (h=0.99 + delta → clamps to 1.0)]

:success-criteria
pass[All 6 golden scenarios pass regression tests] pass[Tau floors and caps behave as documented] pass[Hunger saturation clamps correctly at boundaries] fail[Affective dynamics change without golden update]

:psr-example ” Selected hunger goldens for refactor safety because affect dynamics are most sensitive to drift”
:pur-template ” Hunger scenario {{name}}, h: {{h_start}}→{{h_end}}, tau: {{tau}}”

! instantiated-by: Prototype 4 — Policy Layer Harness [🍲/乡 🔥/代]
:maturity :active
:depends-on [f2/P0, f2/P2, f2/P3]
:evidence-for-active [5 golden scenarios locked; deterministic eval-policy function operational]
+context:ants.aif.policy selects actions using EFE metrics.
+if:Policy choices must be reproducible under controlled inputs
+however:Without a standalone harness, policy testing requires full sim runs
+then:Provide eval-policy harness that accepts synthetic mu/observation pairs; lock golden action rankings for canonical scenarios
+because:Stabilised policies are mandatory before benchmarking or bridging into futon1 salience services
+evidence: evidence[test/resources/goldens/policy_harness.edn has 5 golden scenarios with action rankings] evidence[tools/policy_harness.clj generates canonical scenarios] evidence[test/ants/aif/policy_harness_test.clj compares rankings to golden] evidence[test/ants/aif/policy_test.clj has

:success-criteria
pass[check! returns structured verdicts (applies/blocked/missing)] pass[Every check logged as proof-state record] pass[Schema validation enforced on inputs] fail[Checks proceed without logging]

:psr-example ” Selected check DSL for pattern evaluation because executable checks enable automation”
:pur-template ” Check {{pattern}}, context: {{ctx}}, result: {{status}}”

! instantiated-by: Prototype 3 — Trail & Proof-State Journal [➰/双]
:maturity :active
:depends-on [f3/P2]
:evidence-for-active [Tatami sessions captured; trail schema operational; cue embedding tested]
+context:Wisdom trails document proof obligations and how work sessions discharged them.
+if:You want every check, dismissal, and pattern idea to leave an auditable trail
+however:Without proof-state schema, trails are general exploration logs
+then:Extend trail schema with :pattern/id, :obligation/id, :action/tags, :delta/joy; add rollups showing which devmap clauses advanced per day; expose /musn/trails/proof export
+because:The whole point is to measure how daily practice advances the devmaps
+evidence: evidence[resources/tatami-events.edn captures live tatami sessions] evidence[resources/tatami-context.edn logs FROM-CHATGPT-EDN payloads] evidence[test/futon3/cue_embedding_test.clj proves keyword payloads round-trip] evidence[Tatami HUD mirrors entries through /musn/cues]

+ NEXT-STEPS:
next[Persist cue annotations into trail export] next[Add proof artifacts (HUD screenshots or /musn/cues transcripts)] next[Implement daily rollups of clause advancement]

:success-criteria
pass[Trails include pattern/obligation IDs] pass[Rollups show devmap clause progress per day] pass[Proof export available for futon4 archives] fail[Proof states not captured in trails]

:psr-example ” Selected trails for proof journaling because auditable records enable reflection”
:pur-template ” Trail {{session}}, patterns: {{count}}, obligations cleared: {{cleared}}”

! instantiated-by: Prototype 4 — Workday Instrumentation [➰/止 ⌛/了]
:maturity :active
:depends-on [f3/P2]
:evidence-for-active [workday/submit endpoint operational; aob-chatgpt bridge working]
+context:aob-chatgpt and fubar.el instrument user interactions for pattern checking.
+if:You want FUTON3 to serve as backend for daily reflections
+however:Without official API, workday claims cannot flow through check pipeline
+then:Publish workday/submit endpoint (timestamp, activity, evidence links); run through check DSL + embedding; respond with pattern hits/misses and follow-up obligations
+because:This is how empowered action becomes measurable proof progress
+evidence: evidence[src/futon3/workday.clj implements workday intake] evidence[src/f2/transport.clj exposes /musn/workday/submit HTTP endpoint] evidence[docs/api/musn.yaml documents /musn/workday/submit] evidence[test/transport_test.clj covers workday submission] evidence[README documents workday API]

+ NEXT-STEPS:
next[Replace futon3/logs/workday.edn queue with futon1 persistence (workday->graph)] next[Integrate fubar.el alongside aob-chatgpt]

:success-criteria

pass[Toggle commands differentiate derivative types] pass[Main article buffer shows overlays] fail[Inclusion/transclusion indistinguishable]

:psr-example ” Selected scholium mode for document relationships because visual previews aid navigation”
:pur-template ” Scholium view, inclusions: {{inc}}, transclusions: {{trans}}, identifications: {{id}}”

! instantiated-by: Prototype 3 — Design Pattern Templates [🙍/双 💬/卡]
:maturity :active
:depends-on [f4/P1]
:evidence-for-active [Flexiarg format operational; pattern templates translate to graph nodes]
+context:Design patterns in a standard template need to be editable as both text and graph.
+if:You want patterns to be human-readable and machine-structured
+however:Without a translation layer, patterns stay as flat text
+then:Define schema mapping argument blocks (IF/HOWEVER/THEN/BECAUSE) to graph nodes; support bidirectional editing
+because:Bidirectional editing lets humans and machines work on the same artefacts
+evidence: evidence[Flexiarg format documented and in use across devmaps] evidence[Pattern templates parseable as structured blocks]

+ NEXT-STEPS:
next[Implement graph node generation from flexiarg blocks] next[Add bidirectional sync (graph changes reflect in text)] next[Test round-trip editing of pattern templates]

:success-criteria
pass[Flexiarg blocks translate to graph nodes] pass[Graph edits reflect back in text] pass[Pattern metadata preserved in translation] fail[Translation loses structure or content]

:psr-example ” Selected pattern templates for bidirectional editing because patterns serve humans and machines”
:pur-template ” Pattern translate {{id}}, nodes: {{count}}, round-trip: {{pass|fail}}”

! instantiated-by: Prototype 4 — Graph Storage Layer [🙍/久 👀/乃]
:maturity :greenfield
:depends-on [f4/P1]
:evidence-for-active [Storage wired to Datascript; nodes addressable and queryable]
+context:Futon4 stores text fragments and hyperlinks as addressable nodes.
+if:You want fragments to be queryable and linkable
+however:No memory manager wires Arxana buffers to Datascript
+then:Implement add-node, activate-node, link-node; persist via Datascript; replay sessions to prove round-trips
+because:Without a living graph memory, Arxana cannot escape editor mode
+evidence: evidence[Arxana store talks to futon1 API] evidence[Basic node operations exist]

+ NEXT-STEPS:
next[Wire Arxana buffers to Datascript] next[Implement add-node, activate-node, link-node] next[Test session replay for round-trip verification]

:success-criteria
pass[Nodes stored in Datascript] pass[Nodes queryable by ID and content] pass[Session replay reconstructs state] fail[Node operations not persisted]

:psr-example ” Selected graph storage for queryable fragments because addressable nodes enable linking”
:pur-template ” Node operation {{op}}, node: {{id}}, persisted: {{yes|no}}”

! instantiated-by: Prototype 5 — XTDB Version History [⌛/久 🙍/分]

implements CA mechanics] evidence[src/futon5/wiring/runtime.clj executes wiring diagrams as CA rules] evidence[scripts/hexagram_wiring_sheet.clj generates test sheet for 64 hexagram wirings] evidence[Pattern failure detectors (BARCODE, CANDYCANE) implemented]

+ NEXT-STEPS:
next[Expand wiring diagram vocabulary] next[Add more failure detectors] next[Document MetaCA semantics]

:success-criteria
pass[Wiring diagrams execute as CA rules] pass[Pattern evolution observable] pass[Failure modes detectable] fail[Diagrams do not execute]

:psr-example ” Selected MetaCA as demonstrator because CA provides executable pattern semantics”
:pur-template ” MetaCA run {{id}}, generations: {{gen}}, entropy: {{entropy}}, failures: {{fails}}”

! instantiated-by: Prototype 4 — Transfer Milestone [🔺/另 ➰/一]
:maturity :settled
:depends-on [f5/P2, f5/P3]
:evidence-for-settled [Wiring diagrams transfer from MetaCA to ant simulation in futon2]
+context:The transfer milestone proves wiring diagrams developed in one domain work in another.
+if:You want confidence that patterns are genuinely transferable
+however:Transfer claims need demonstration, not just assertion
+then:Develop wiring diagrams in MetaCA, apply them to ant simulation in futon2, document results
+because:Proven transfer validates the pattern approach
+evidence: evidence[Wiring diagrams from MetaCA applied to futon2 ants] evidence[Same diagram semantics work in both domains] evidence[futon2 AIF engine can consume futon5 patterns]

+ NEXT-STEPS:
next[Document additional transfer examples] next[Test transfer to music domain]

:success-criteria
pass[Diagrams developed in MetaCA work in ants] pass[Transfer documented with evidence] pass[Semantics preserved across domains] fail[Transfer requires domain-specific changes]

:psr-example ” Selected transfer milestone for validation because transfer proves pattern generality”
:pur-template ” Transfer {{pattern}}, source: {{src}}, target: {{tgt}}, result: {{pass|fail}}”

! instantiated-by: Prototype 5 — Dynamic Programming for Action Sequencing [🎶/己 🌲/今]
:maturity :greenfield
:depends-on [f5/P2]
:evidence-for-active [DP implemented for action sequencing; music domain operational]
+context:Dynamic programming manages sequencing of actions in demonstration domains including music.
+if:You want optimal or near-optimal action sequences in improvisation
+however:Naive sequencing ignores structure and dependencies
+then:Use dynamic programming to compute action sequences; apply to music and other demonstration domains
+because:DP provides principled sequencing that respects constraints
+evidence: evidence[DP concepts documented] evidence[Music domain identified as target]

+ NEXT-STEPS:
next[Implement DP for action sequencing] next[Apply to music improvisation] next[Test on other domains]

field: {{domain}}, usages: {{count}}”

! instantiated-by: Prototype 4 — Reasoning Map [👉/田 🔺/今]
:maturity :greenfield
:depends-on [f6/P0, f6/P2, f6/P3]
:evidence-for-active [Reasoning map connects informal, cross-domain, and domain-specific patterns]
+context:Three kinds of pattern need to connect coherently.
+if:You want to see how informal, cross-domain, and domain-specific patterns work together
+however:Without a map, the pattern layers remain isolated
+then:Build a reasoning map showing how the three pattern types fit together to make mathematics work
+because:The map reveals the structure of mathematical reasoning
+evidence: evidence[Three pattern types defined] evidence[Seed domain provides test cases]

+ NEXT-STEPS:
next[Design reasoning map structure] next[Populate with seed domain examples] next[Test navigation across pattern types]

:success-criteria
pass[Map connects all three pattern types] pass[Navigation works across levels] pass[Structure visible to users] fail[Pattern types remain isolated]

:psr-example ” Selected reasoning map for coherence because patterns must connect”
:pur-template ” Map query {{entry}}, informal: {{inf}}, cross-domain: {{cd}}, domain: {{dom}}”

! instantiated-by: Prototype 5 — Interactive Tutorials [🌀/习 🐴/二]
:maturity :greenfield
:depends-on [f6/P1, f6/P4]
:evidence-for-active [Tutorials operational; users practice by working problems]
+context:Mathematics is learned by doing, not just reading.
+if:You want people to practice mathematical reasoning
+however:Static entries do not support practice
+then:Build interactive tutorials that let people work through problems using the dictionary content
+because:Practice builds fluency
+evidence: evidence[Seed domain provides tutorial content] evidence[F3 training ground provides model]

+ NEXT-STEPS:
next[Design tutorial format] next[Create tutorials for seed domain] next[Test with users]

:success-criteria
pass[Tutorials guide problem-solving] pass[Users can practice reasoning moves] pass[Progress tracked] fail[Tutorials passive or unusable]

:psr-example ” Selected interactive tutorials for learning because practice builds fluency”
:pur-template ” Tutorial {{id}}, problems: {{count}}, completed: {{done}}”

! instantiated-by: Prototype 6 — Landscape Mode [🙍/无 🐲/乃]
:maturity :greenfield
:depends-on [f6/P1, f6/P4]
:evidence-for-active [Landscape view shows connected neighbourhoods]
+context:Mathematical entries have neighbours: related lemmas, examples, counterexamples.
+if:You want to navigate mathematics as connected terrain
+however:Flat lists hide structure
+then:Build landscape mode that renders neighbourhoods of related content
+because:Connected views reveal mathematical structure
+evidence: evidence[Seed domain entries have cross-references] evidence[F1 graph memory

from HUD] pass[Simulations are runnable without context switch] fail[Transparency tools remain siloed from main interface]

! instantiated-by: Completion — Full Layer Integration [🌽/末 🌏/万]
:maturity :greenfield
:depends-on [f0/P0, f0/P1, f0/P2, f0/P3, f0/P4, f0/P5, f0/P6, f0/P7]
+context:F0 is complete when all layer interfaces work.
+if:P0-P7 are all operational
+however:Integration may reveal gaps not visible in individual prototypes
+then:Validate that the human can access all stack capabilities through F0
+because:The system extends human capability only if all layers are accessible

+ NEXT-STEPS:
next[Integration test: access all layers through morning review workflow] next[Identify friction points in layer transitions] next[Validate that capability extension is real, not theoretical]

:success-criteria
pass[Human can access memory, agents, verification, hypertext, patterns, math, transparency through unified interface] pass[Layer transitions are fluid, not jarring] fail[Some layers remain inaccessible or require context switch]

output varies non-deterministically]

:psr-example ” Selected nlp-interface for text processing because deterministic tagging is foundational”
:pur-template ” NLP processed {{char-count}} chars, entities: {{entity-count}}, deterministic: {{yes|no}}”

! instantiated-by: Prototype 5 — Open-World Ingest [📥/入 🌏/开]
:maturity :active
:depends-on [f1/P3, f1/P4]
:evidence-for-settled [Ingest outputs stable across 20+ corpus runs; ID stability verified]
+context:Open-world ingest accumulates background knowledge.
+if:FUTON1 should support both interactive exploration and offline corpus ingestion
+however:Ingest and interactive workflows must share the same persistent state coherently
+then:Stream text through deterministic CoreNLP pipeline, write entities/relations to XTDB, share state with demo/client
+because:Background knowledge should accumulate durably without breaking interactive exploration
+evidence: evidence[apps/open-world-ingest streams text through CoreNLP to XTDB] evidence[Demo/client workflows hydrate from same XTDB and see ingest-produced documents] evidence[Open-world invariants pass end-to-end] evidence[README confirms deterministic ingest via CoreNLP and XTDB mirroring]

+ NEXT-STEPS:
next[Define standard data-root layout for ingest and demo/client] next[Document how ingest-produced documents appear in hydrated store] next[Add golden tests for entity-ID stability and relation formation] next[Provide ingest→query example workflow]

:success-criteria
pass[Ingest produces stable entity IDs across runs] pass[Ingest-produced documents visible in demo/client after hydration] pass[Questions handled correctly (not treated as assertions)] fail[Ingest corrupts or conflicts with interactive state]

:psr-example ” Selected open-world-ingest for corpus loading because background knowledge needs durable storage”
:pur-template ” Ingested {{doc-count}} documents, entities: {{entity-count}}, relations: {{rel-count}}”

! instantiated-by: Prototype 6 — Demo/Client (API Demonstration) [💖/业 📁/示]
:maturity :active
:depends-on [f1/P0]
:evidence-for-settled [Demo exactly mirrors API behaviour; used as API reference in documentation]
+context:The demo/client workflow demonstrates the API.
+if:Developers need to understand what the API does
+however:API documentation alone may not convey behaviour clearly
+then:Ensure demo/client is an exact demonstration of API capabilities—same inputs produce same outputs
+because:A faithful demo serves as executable documentation for the canonical interface
+evidence: evidence[apps/demo and apps/client exercise the API through command-service] evidence[Focus headers emitted identically via CLI and API] evidence[README documents demo/client as API demonstration]

+ NEXT-STEPS:
next[Verify demo/client outputs match API responses exactly] next[Add demo transcript that serves as API example] next[Document any demo-specific behaviour that differs from API]

:success-criteria
pass[Demo produces identical output to equivalent API calls] pass[Demo serves as executable API documentation] fail[Demo behaviour

comprehensive behavioral tests]

+ NEXT-STEPS:
next[Expand to 10 canonical scenarios as specified] next[Document EFE component formulas in doc/policy-spec.md] next[Add policy response time instrumentation]

:success-criteria
pass[eval-policy produces deterministic rankings for same inputs] pass[Golden scenarios cover forage, return, explore, defend modes] pass[Policy changes require golden updates with rationale] fail[Action rankings vary non-deterministically]

:psr-example ” Selected policy harness for fulab agent validation because reproducible decisions enable debugging”
:pur-template ” Policy eval scenario {{name}}, top action: {{action}}, G: {{G}}”

! instantiated-by: Prototype 5 — World Mechanics Contract [🎑/书 👕/办]
:maturity :greenfield
:depends-on []
:evidence-for-active [Contract documented; invariants tested; movement deterministic]
+context:The simulator handles evaporation, movement, gather/deposit, pheromone dynamics, hunger propagation, and reserves.
+if:FUTON2 should be a reliable agent-environment backend
+however:Movement heuristics and gather thresholds are powerful yet lightly documented
+then:Document all invariants (pheromone caps, grid bounds, gather limits), seed movement for determinism, and split movement modes into testable units
+because:A clear contract keeps the world trustworthy when other prototypes depend on it for episodes
+evidence: evidence[war.clj implements full world mechanics] evidence[Basic simulation runs deterministically with fixed seeds]

+ NEXT-STEPS:
next[Document pheromone caps, grid bounds, gather limits in doc/world-contract.md] next[Add property tests for movement determinism] next[Split movement modes into testable units] next[Document white-streak rules]

:success-criteria
pass[World invariants documented and tested] pass[Movement deterministic for same seed] pass[Gather/deposit thresholds explicit and tested] fail[World state drifts non-deterministically]

:psr-example ” Selected world contract for episode export because external consumers need guaranteed semantics”
:pur-template ” World step {{tick}}, ants: {{count}}, food: {{food}}, pheromone max: {{pher}}”

! instantiated-by: Prototype 6 — Pivot Stream & Analyzer [🚢/已 🐲/无]
:maturity :greenfield
:depends-on [f2/P0]
:evidence-for-active [Schema versioned; required keys tested; futon3/futon4 can subscribe]
+context:The live analyzer consumes pivot events, computes EWMA metrics, and detects starvation spirals.
+if:Analyzer outputs should feed futon3 evaluations or futon4 reflexive agents
+however:Pivot schemas are implicit and analyzer assumptions undocumented
+then:Freeze versioned pivot-row schema, test required keys, and ship example analyzer plugin
+because:Pivot streams are FUTON2’s event bus and must behave like a stable ABI
+evidence: evidence[Live analyzer operational in war.clj] evidence[Pivot events emitted per tick]

+ NEXT-STEPS:
next[Freeze pivot-row schema in doc/pivot-schema.md] next[Add tests for required keys (:act, :mode, :cargo, :ing, etc.)] next[Ship example analyzer plugin for futon4] next[Document EWMA metric calculations]

pass[Workday claims flow through check DSL] pass[Pattern hits/misses returned with follow-up obligations] pass[Both aob-chatgpt and fubar.el can submit] fail[Workday claims bypass pattern checking]

:psr-example ” Selected workday instrumentation for aob-chatgpt because daily practice needs pattern feedback”
:pur-template ” Workday submit {{timestamp}}, patterns matched: {{count}}, obligations: {{new}}”

! instantiated-by: Prototype 5 — Similarity Field [🌂/甲 👂/归]
:maturity :active
:depends-on [f3/P1]
:evidence-for-active [futon3a provides pattern indexing with GloVe/fastText/MiniLM embeddings; portal queries operational]
+context:Sigil-distance and GloVe embeddings enable pattern suggestions.
+if:You want reproducible neighbourhood queries for pattern suggestions
+however:Embeddings live in futon3a, not wired into FUTON3 APIs
+then:Integrate sigil-distance matrix and GloVe embeddings as first-class service (pattern search API, nearest-patterns command, CLI/UI views); publish tests proving deterministic neighbourhoods
+because:Pattern suggestion quality underpins the flexiformal workflow
+evidence: evidence[Sigil-distance matrix exists in CSVs] evidence[GloVe embeddings available] evidence[futon3a/scripts/index_patterns.sh builds notions index with GloVe/fastText/MiniLM] evidence[futon3a/scripts/portal provides pattern search and suggest commands] evidence[futon3a/docs/pattern-indexing.md documents embedding pipeline] evidence[src/futon3/pattern_hints.clj exposes nearest-patterns helper] evidence[scripts/nearest-patterns provides nearest-patterns command] evidence[fucodex/fuclaude route nearest-patterns subcommand]

+ NEXT-STEPS:
next[Integrate futon3a portal into futon3 HUD server for pattern suggestions] next[Wire futon3a embeddings into futon3 pattern search API] next[Publish deterministic neighbourhood tests] next[Unify futon3 and futon3a pattern stores or establish clear bridge]

:success-criteria
pass[Pattern search API returns nearest patterns] pass[Neighbourhoods deterministic across runs] pass[Both sigil-distance and GloVe queryable] fail[Pattern suggestions require manual lookup]

:psr-example ” Selected similarity field for pattern discovery because neighbourhood queries enable exploration”
:pur-template ” Similarity query {{pattern}}, neighbours: {{count}}, method: {{sigil|glove}}”

! instantiated-by: Prototype 6 — Intent Embedding & Proof Typing [➰/双 🍍/办]
:maturity :greenfield
:depends-on [f3/P3]
:evidence-for-active [Intent→vec pipeline operational; fruit/orb types assigned to proof steps]
+context:Trails record patterns but don’t type proof steps as constructive operations.
+if:You want to distinguish ” this is a check” from ” this is an indicator preserving a virtue invariant”
+however:Current schema has no machinery to interpret intent→pattern reasoning as typed proof steps
+then:Define intent→vec pipeline mapping :trail/intent + :pattern/reasoning to (fruit, orb) pairs where fruits encode operator classes and orbs encode meta-properties; compute :rule/salience; write back to trail events
+because:In Kolmogorov’s sense, a proof is a construction; typing steps like ” indicator→virtue-aligned” lets you see what kind of proof work actually happened
+evidence: evidence[Cue embedding tests show keyword payloads round-trip] evidence[resources/paramita-grid.edn provides gold set for

:maturity :greenfield
:depends-on [f4/P4]
:evidence-for-active [XTDB integrated; time-travel queries working; branching histories supported]
+context:XTDB adds version history so you can see how documents evolved.
+if:You want durability, branching histories, and provenance queries
+however:XTDB idioms differ from Datascript and require schema migration
+then:Rekey schema for XTDB, add transaction logs, implement time-travel queries
+because:Version history turns Arxana into a serious memory engine
+evidence: evidence[XTDB available in futon1 stack] evidence[Schema migration patterns documented]

+ NEXT-STEPS:
next[Design XTDB schema for Arxana nodes] next[Implement transaction logging] next[Add time-travel query interface] next[Test branching and merge scenarios]

:success-criteria
pass[Documents have version history] pass[Time-travel queries return past states] pass[Branching histories supported] fail[History lost on edit]

:psr-example ” Selected XTDB for version history because document evolution needs to be traceable”
:pur-template ” Time-travel query, node: {{id}}, timestamp: {{ts}}, found: {{yes|no}}”

! instantiated-by: Prototype 6 — Multi-User Support [🔦/门 🙅/互]
:maturity :greenfield
:depends-on [f4/P5]
:evidence-for-active [Multi-user sync operational; collaborators share semantic space]
+context:Multi-user support lets collaborators share a semantic space.
+if:You want multiple people to work in the same document graph
+however:No multi-user or sync layer exists yet
+then:Use Datascript local + XTDB remote as stepping stones; implement sync protocol
+because:Shared memory opens the door to collaborative reasoning
+evidence: evidence[XTDB provides remote persistence foundation] evidence[Datascript provides local cache]

+ NEXT-STEPS:
next[Design sync protocol between local and remote] next[Implement conflict resolution] next[Test multi-user editing scenarios] next[Add presence indicators]

:success-criteria
pass[Multiple users see consistent state] pass[Edits propagate to all users] pass[Conflicts resolved gracefully] fail[Users see divergent states]

:psr-example ” Selected multi-user support for collaboration because shared memory enables co-reasoning”
:pur-template ” Sync event, users: {{count}}, conflicts: {{conflicts}}, resolved: {{yes|no}}”

! instantiated-by: Prototype 7 — Literary Interface [🙅/也 💖/久]
:maturity :greenfield
:depends-on [f4/P2]
:evidence-for-active [Manuscript mode operational; chunking and cross-reference working]
+context:A literary interface supports long-form writing like books and manuscripts.
+if:You want to edit books or long-form text via Arxana
+however:No good UI for long-form text exists yet
+then:Add manuscript mode with chunking and cross-reference map
+because:FUTON becomes a writing environment, not just an editor
+evidence: evidence[Arxana handles document fragments] evidence[Scholium mode provides

:success-criteria
pass[DP produces valid action sequences] pass[Music domain operational] pass[Sequencing respects constraints] fail[Sequences violate domain constraints]

:psr-example ” Selected dynamic programming for sequencing because DP respects structure”
:pur-template ” DP sequence {{domain}}, actions: {{count}}, optimal: {{yes|no}}”

provides storage]

+ NEXT-STEPS:
next[Design neighbourhood visualization] next[Implement for seed domain] next[Test navigation]

:success-criteria
pass[Neighbourhoods render correctly] pass[Related content visible] pass[Navigation intuitive] fail[Entries appear isolated]

:psr-example ” Selected landscape mode for navigation because connected views reveal structure”
:pur-template ” Landscape {{entry}}, neighbours: {{count}}, depth: {{levels}}”

! instantiated-by: Prototype 7 — StackExchange Import [🙅/叉 💤/久]
:maturity :greenfield
:depends-on [f6/P0]
:evidence-for-active [Q&A pairs imported; mapped to dictionary schema]
+context:StackExchange has thousands of mathematical Q&A pairs.
+if:You want to seed the dictionary with real content
+however:External content needs mapping to our schema
+then:Build importer that maps StackExchange Q&A to dictionary entries with pattern tags
+because:Curated Q&A provides initial density
+evidence: evidence[StackExchange Math data available] evidence[Schema defined for entries]

+ NEXT-STEPS:
next[Build StackExchange parser] next[Map Q&A to entry schema] next[Tag with patterns] next[Review imported content]

:success-criteria
pass[Q&A pairs import correctly] pass[Mapped to dictionary schema] pass[Pattern tags applied] fail[Import corrupts or loses content]

:psr-example ” Selected StackExchange import for seeding because real Q&A provides density”
:pur-template ” Import batch {{id}}, questions: {{count}}, tagged: {{tagged}}”

! instantiated-by: Prototype 8 — Agent Protocol [🙅/无 🍍/工]
:maturity :greenfield
:depends-on [f6/P1]
:evidence-for-active [Protocol defined; agents query and edit safely]
+context:Agents will query, annotate, and refactor dictionary entries.
+if:You want safe and productive agent behaviour
+however:Unrestricted access risks corruption
+then:Define protocol specifying allowed queries, acceptable edits, and review requirements
+because:Bounded access keeps content safe while enabling automation
+evidence: evidence[F3 agent protocol provides model] evidence[Dictionary schema defines valid operations]

+ NEXT-STEPS:
next[Define allowed queries] next[Define acceptable edits] next[Implement review workflow] next[Test with agents]

:success-criteria
pass[Agents query without errors] pass[Edits reviewed before commit] pass[No unauthorized changes] fail[Agents corrupt or bypass review]

:psr-example ” Selected agent protocol for safety because bounded access enables automation”
:pur-template ” Agent action {{type}}, query: {{query}}, reviewed: {{yes|no}}”

! instantiated-by: Prototype 9 — Reusable Mathematical Models [🔸/小 🐊/十]
:maturity :greenfield
:depends-on [f6/P1, f6/P2]
:evidence-for-active [Models describe F6 development; reusable in other domains]
+context:General purpose mathematical models

diverges from API without documentation]

:psr-example ” Selected demo/client for API learning because executable examples clarify behaviour”
:pur-template ” Demo session {{session-id}}, API-equivalent: {{yes|no}}, divergences: {{count}}”

! instantiated-by: Prototype 7 — Query Workflows [👓/见 ✌️/见]
:maturity :greenfield
:depends-on [f1/P0, f1/P3]
:source [futon3a/P0]
:evidence-for-active [Query functions packaged and documented; used in 10+ inspection sessions]
+context:The stored graph must be inspectable through predictable queries.
+if:Developers need to examine entities, relations, salience, and schema
+however:Common inspection tasks currently require manual XTDB queries or scattered snippets
+then:Package reusable query functions (list-entities, list-relations, inspect-entity, counts) with examples
+because:Consolidated query workflows reduce friction and strengthen FUTON1 as inspectable substrate
+evidence: evidence[-M:storage/inspect provides automated query examples] evidence[README-storage.md documents XTDB inspection behaviour] evidence[futon3a/P0 Portal developing complementary pattern-guided retrieval]

+ NEXT-STEPS:
next[Package query functions with brief examples in one place] next[Cross-reference futon3a Portal for pattern-guided queries] next[Surface XTDB schema in Emacs for futon3 browsing] next[Add query latency instrumentation]

:success-criteria
pass[Common queries available as named functions with examples] pass[Query workflows consistent between REPL and Emacs views] pass[futon3a Portal can use FUTON1 queries as backend] fail[Inspection requires ad-hoc XTDB query construction]

:psr-example ” Selected query-workflows for graph inspection because predictable queries reduce friction”
:pur-template ” Query {{query-type}}, results: {{count}}, latency: {{ms}}ms”

! instantiated-by: Prototype 8 — Pilot Storage (fulab, MMCA) [🎑/正 🍵/内]
:maturity :greenfield
:depends-on [f1/P3, f1/P1]
:source [futon5/E1 fulab/experiment-runs]
:evidence-for-active [fulab runs stored and retrievable; MMCA experiments stored with replay capability]
+context:Experimental workflows need reproducible storage.
+if:fulab runs and MMCA experiments need durable, replayable state
+however:No standard way to package pilot configuration, transcripts, and expected state
+then:Document pilot assembly using FUTON1 tools; implement storage for fulab (partly done) and MMCA experiments (planned)
+because:Experimental futons (futon2, futon5) require stable backend for storing and replaying world state
+evidence: evidence[fulab runs partly implemented with FUTON1 storage] evidence[MMCA experiment storage planned for futon5 E1] evidence[FUTON1 supports deterministic transcripts and XTDB-backed persistence] evidence[CLI and API can drive scripted workflows]

+ NEXT-STEPS:
next[Complete fulab run storage integration] next[Design MMCA experiment storage schema] next[Document pilot assembly guide with worked example] next[Provide minimal pilot example with config, transcripts, expected state]

:success-criteria
pass[Pivot schema versioned and stable] pass[Required keys validated on emit] pass[External consumers can subscribe without internal knowledge] fail[Schema changes break downstream consumers]

:psr-example ” Selected pivot schema for futon4 integration because event bus stability enables composition”
:pur-template ” Pivot batch {{batch}}, events: {{count}}, schema version: {{version}}”

! instantiated-by: Prototype 7 — External Stepping API [📤/予 🚢/已]
:maturity :greenfield
:depends-on [f2/P5, f2/P6]
:evidence-for-active [futon2.step function exposed; EDN episode export working]
+context:simulate drives the war sim but offers no external stepping API.
+if:futon3 fulab or futon5 MetaCA want to request ” give me N steps with this config”
+however:Stepping remains tied to CLI and HUD
+then:Expose (futon2.step world n {:emit-pivot? true}), add EDN exports of full episodes
+because:Exported episodes are critical for agent training, memory, and pattern extraction
+evidence: evidence[simulate function exists in war.clj] evidence[Basic stepping possible via REPL]

+ NEXT-STEPS:
next[Expose futon2.step as public API] next[Add EDN episode export with pivot batches] next[Document stepping API for futon3/futon5 consumers] next[Add episode replay capability]

:success-criteria
pass[External callers can step world programmatically] pass[Episodes exportable as EDN] pass[Pivot batches consistent across export] fail[Stepping requires internal knowledge or HUD]

:psr-example ” Selected stepping API for fulab because agents need programmatic episode control”
:pur-template ” Episode export {{id}}, steps: {{count}}, pivots: {{pivot_count}}”

! instantiated-by: Prototype 8 — Benchmark Suite [🗿/三 ⚡/内]
:maturity :greenfield
:depends-on [f2/P7]
:evidence-for-active [Parametrised benchmarks across grid sizes; golden CSVs captured]
+context:Benchmark scripts compare classic ants to AIF runs across scores and durations.
+if:You want formal behavioural comparisons rather than anecdotes
+however:Only three preset scenarios exist with no variance analysis
+then:Add parametrised benchmarks spanning grid sizes, pheromone decay, hunger burn, and tau floors; capture plots and golden CSVs
+because:FUTON2 should function as a scientific system with reproducible metrics
+evidence: evidence[Basic benchmark scripts exist] evidence[Classic vs AIF comparison possible]

+ NEXT-STEPS:
next[Parametrise benchmarks across grid sizes, decay rates, hunger burn] next[Add variance analysis across seeds] next[Capture golden CSVs for regression] next[Generate comparison plots]

:success-criteria
pass[Benchmarks parametrised and reproducible] pass[Classic vs AIF differences quantified] pass[Golden CSVs prevent silent regressions] fail[Benchmark results vary unexpectedly]

:psr-example ” Selected benchmark suite for AIF validation because claims need quantified evidence”
:pur-template ” Benchmark {{name}}, classic score: {{classic}}, AIF score: {{aif}}, delta: {{delta}}”

fruit/orb recovery]

+ NEXT-STEPS:
next[Generalise cue embedding to operate on intent + reasoning tuples] next[Calibrate fruit/orb basis on manually typed patterns] next[Wire enriched trail export into rollups] next[Add HUD affordances to filter by proof type]

:success-criteria
pass[Each trail step typed as (fruit, orb) pair] pass[Salience scores computed for proof steps] pass[Rollups show proof types that fired per day] fail[Proof steps remain untyped]

:psr-example ” Selected intent embedding for proof typing because Kolmogorov-style types enable meta-reasoning”
:pur-template ” Intent embed {{step}}, fruit: {{fruit}}, orb: {{orb}}, salience: {{score}}”

! instantiated-by: Prototype 7 — Pattern Creation Workbench [👍/已]
:maturity :greenfield
:depends-on [f3/P3, f3/P6]
:evidence-for-active [Summariser clusters trails; draft flexiargs emitted with readiness criteria]
+context:New patterns emerge from trails and devmap edits but capture is manual.
+if:You want FUTON3 to draft candidate patterns from accumulated proof states
+however:Manual backlog creates tension; no readiness criteria for handoff
+then:Build summariser that clusters trails, extracts hooks, emits draft flexiarg blocks (IF/HOWEVER/THEN/BECAUSE) with suggested pāramitā; declare readiness criteria (N supporting checks + steward review)
+because:FUTON3 should propose expansions grounded in lived work
+evidence: evidence[Trail data available for clustering] evidence[Flexiarg format documented]

+ NEXT-STEPS:
next[Build trail clustering summariser] next[Emit draft flexiarg blocks with pāramitā tags] next[Define readiness criteria for handoff to futon4] next[Add steward review workflow]

:success-criteria
pass[Summariser emits draft flexiargs from trails] pass[Drafts tagged with suggested pāramitā] pass[Readiness criteria declared and enforced] fail[Pattern creation remains fully manual]

:psr-example ” Selected pattern workbench for library growth because automated drafts reduce manual burden”
:pur-template ” Workbench draft {{id}}, sources: {{trail_count}}, readiness: {{status}}”

! instantiated-by: Prototype 8 — Joy Metrics [⌛/力 ⚡/二]
:maturity :greenfield
:depends-on [f3/P3, f3/P4]
:evidence-for-active [Joy metrics computed per session; provided to futon0]
+context:Joy = Spinozan increase in power to act, evidenced by commitments you can now keep.
+if:You want metrics showing whether checks are empowering or draining
+however:Current metrics counted UI friction, not proofs discharged
+then:Track time-to-proof, obligations-cleared, follow-on commitments, viriya linkages per session; visualise alongside subjective notes; provide to futon0
+because:Empowered action is the KPI for this futon
+evidence: evidence[Session data available from trails] evidence[Obligation tracking in check DSL]

+ NEXT-STEPS:
next[Define joy metric formulas] next[Implement per-session computation] next[Wire metrics to futon0] next[Add visualization alongside subjective notes]

:success-criteria
pass[Joy metrics computed per session]

linking infrastructure]

+ NEXT-STEPS:
next[Design manuscript mode UI] next[Implement chapter/section chunking] next[Add cross-reference map visualization] next[Test with real manuscript content]

:success-criteria
pass[Long documents chunked into navigable sections] pass[Cross-references visualized and navigable] pass[Manuscript mode supports book-length content] fail[Long documents unwieldy to edit]

:psr-example ” Selected literary interface for book editing because long-form writing needs specialized support”
:pur-template ” Manuscript session, chapters: {{count}}, cross-refs: {{refs}}, words: {{words}}”

should be reusable.
+if:You want models that transfer to other knowledge domains
+however:Models are often locked to their original context
+then:Use mathematical models to describe F6’ s own growth and development; present them for reuse elsewhere
+because:Eating your own cooking validates reusability
+evidence: evidence[F6 development provides test case] evidence[Growth/equilibrium models identified]

+ NEXT-STEPS:
next[Identify models applicable to F6 development] next[Document models in reusable form] next[Test application to other domains]

:success-criteria
pass[Models describe F6 development] pass[Models documented for reuse] pass[At least one external application] fail[Models locked to F6 context]

:psr-example ” Selected reusable models for transfer because general models should work elsewhere”
:pur-template ” Model {{id}}, domain: {{F6|other}}, reused: {{yes|no}}”

:success-criteria
pass[fulab runs stored and retrievable via FUTON1] pass[MMCA experiments can be stored and replayed] pass[Pilot state verifiable through query workflows] fail[Experiment state lost or non-reproducible]

:psr-example ” Selected pilot-storage for fulab integration because experiments need durable replay”
:pur-template ” Stored pilot {{pilot-id}}, type: {{fulab|mmca}}, artifacts: {{count}}”

! instantiated-by: Prototype 9 — futon1 Graph Adapter [👎/刊 👭/二]
:maturity :greenfield
:depends-on [f2/P6, f2/P7]
:source [f1/P7 f1/P8]
:evidence-for-active [Adapter converts percepts↔graph facts; colony run exports to futon1 salience]
+context:FUTON1 offers graph-memory API; FUTON2 emits percept/action streams.
+if:Ants should be the first AIF agents that read fact graphs and write beliefs
+however:Percepts/actions are locked to grid coordinates with no mapping to futon1 entity IDs
+then:Design bidirectional adapter (percepts→graph facts, graph cues→priors), version ABI, replay colony run that exports beliefs into futon1
+because:A working bridge proves ants can reason over the same facts as the rest of the stack
+evidence: evidence[futon1 graph-memory API documented and operational] evidence[FUTON2 pivot streams provide raw material for export]

+ NEXT-STEPS:
next[Design percept→entity mapping] next[Design graph cue→prior injection] next[Version adapter ABI] next[Demo: replay colony while futon1 ingests stream]

:success-criteria
pass[Adapter bidirectionally converts percepts and graph facts] pass[Colony beliefs visible in futon1 salience tables] pass[ABI versioned for evolution] fail[Export corrupts futon1 state]

:psr-example ” Selected graph adapter for futon1 integration because shared facts enable cross-futon reasoning”
:pur-template ” Adapter export {{colony}}, entities: {{count}}, relations: {{rels}}”

! instantiated-by: Prototype 10 — futon3 Proofwork Bridge [🌀/内 🙅/刊]
:maturity :greenfield
:depends-on [f2/P6, f2/P7]
:source [f3/trails f3/MUSN]
:evidence-for-active [Trail→mission translator working; agent runs logged as trail payloads]
+context:FUTON3 trails record which patterns were checked during human work sessions.
+if:FUTON2 analyzers should ingest trails and commission agent goals
+however:No ingestion path for trail EDN, no feedback loop to direct agents
+then:Add trail→mission translator, plus reporting showing how agent runs affect subsequent checks
+because:Shared missions keep the energy loop tight between simulations and human proofwork
+evidence: evidence[FUTON3 trails operational with pattern tracking] evidence[FUTON2 can emit mission-style payloads via pivot]

+ NEXT-STEPS:
next[Design trail→mission schema] next[Implement trail ingestion in FUTON2] next[Add mission→agent goal queue] next[Report agent impact on pattern checks]

:success-criteria
pass[Trails ingestible as agent missions] pass[Agent runs produce trail-style payloads] pass[Feedback loop visible in reports] fail[Trails and agent runs disconnected]

:psr-example ” Selected proofwork bridge for shared energy because human and agent work should reinforce”
:pur-template ” Trail ingested {{trail}}, missions: {{count}}, agent runs: {{runs}}”

! instantiated-by: Prototype 11 — Curriculum & Behaviour Cards [🎴/乡 🐜/予]
:maturity :greenfield
:depends-on [f2/P7]
:evidence-for-active [Named curriculum with fixed-seed episodes; behaviour cards exportable]
+context:The current war scenario is rich but single-purpose.
+if:FUTON2 should double as a pedagogical engine for Active Inference
+however:There is no named curriculum

pass[Metrics provided to futon0] pass[Visualizations show empowerment trends] fail[Joy unmeasured or disconnected from futon0]

:psr-example ” Selected joy metrics for empowerment tracking because piti factor requires measurable progress”
:pur-template ” Joy session {{id}}, time-to-proof: {{ms}}, cleared: {{count}}, commitments: {{new}}”

! instantiated-by: Prototype 9 — Proof Hooks (futon1/futon2 Export) [⚡/二 🙅/乃]
:maturity :greenfield
:depends-on [f3/P2, f1/P0, f2/P6]
:evidence-for-active [proof→graph materialises relations; proof→energy emits viriya deltas]
+context:FUTON3 generates proof states; futon1 (graph) and futon2 (viriya) need them as fuel.
+if:You want every accepted check to become a graph relation and viriya delta automatically
+however:Current exports are bespoke scripts
+then:Standardise proof→graph (materialise relations with provenance) and proof→energy (emit activation deltas); lock with fixtures + schemas
+because:Proofs only matter when other futons can act on them
+evidence: evidence[Tatami events available for export] evidence[futon1 graph-memory API operational] evidence[futon2 pivot streams accept deltas]

+ NEXT-STEPS:
next[Wire tatami event log into futon1 via tatami_store.clj] next[Standardise proof→graph adapter with fixtures] next[Standardise proof→energy adapter with fixtures] next[Document export schemas]

:success-criteria
pass[Accepted checks become graph relations automatically] pass[Viriya deltas emitted for futon2 dashboards] pass[Adapters locked with fixtures and schemas] fail[Export requires manual scripting]

:psr-example ” Selected proof hooks for cross-futon integration because automated export enables composition”
:pur-template ” Proof export {{check}}, graph relations: {{rels}}, viriya delta: {{delta}}”

! instantiated-by: Prototype 10 — Flexiformal Training Ground [🚢/氏 🚴/未]
:maturity :greenfield
:depends-on [f3/P2, f3/P3]
:evidence-for-active [Training ground packaged; tutorials complete; simulator recorded]
+context:FUTON3 needs a reproducible training ground for flexiformal practice.
+if:You want collaborators (including FUTON6 math stewards) to experience the proof workflow
+however:No packaged simulator/tutorial bundle exists
+then:Package ” Flexiformal Training Ground” release: tutorials referencing t4r/ patterns, sample check sessions, minimal simulator feeding FUTON6’ s Hyperreal entries; name facilitators and consumers; publish readiness criteria
+because:Training ground turns FUTON3 into a reproducible service for upstream layers
+evidence: evidence[Check DSL operational for training] evidence[Trail capture available for transcripts]

+ NEXT-STEPS:
next[Write tutorials referencing t4r/ patterns] next[Record sample check sessions] next[Build minimal simulator for FUTON6] next[Aggregate proof events into proof-chain.edn]

:success-criteria
pass[Training ground packaged with tutorials] pass[Sample sessions reproducible] pass[Simulator feeds FUTON6 entries] fail[Training requires undocumented setup]

:psr-example ” Selected training ground for onboarding because reproducible practice

+then:Create behaviour cards (home-return, starvation spiral, over-exploration, defensive lattice, greedy gatherer), simulate with fixed seeds, export episodes
+because:These cards seed pattern extraction and future training systems
+evidence: evidence[Fixed-seed simulation possible] evidence[Various behavioral modes observable in runs]

+ NEXT-STEPS:
next[Define 5 canonical behaviour cards with descriptions] next[Record fixed-seed episodes for each] next[Export episodes as training data] next[Document as AIF pedagogy resource]

:success-criteria
pass[Each behaviour card has fixed-seed episode] pass[Episodes exportable for training] pass[Cards serve as AIF teaching examples] fail[Behaviours not reproducible across runs]

:psr-example ” Selected curriculum cards for AIF pedagogy because worked examples accelerate learning”
:pur-template ” Behaviour card {{name}}, seed: {{seed}}, steps: {{steps}}”

! instantiated-by: Prototype 12 — Viriya Metrics [⚡/习 🐜/己]
:maturity :greenfield
:depends-on [f2/P6, f2/P9, f2/P10]
:evidence-for-active [Viriya indicators logged per episode; futon3/futon7 can read energy health]
+context:FUTON2 carries the Factor of Awakening ” energy/viriya” : vigor, commitment, kinetic practice.
+if:Simulations should reflect and inform real social activation
+however:Current metrics stop at hunger/tau; nothing measures vigor or coordination throughput
+then:Introduce viriya indicators (colony activation score, pledge adherence, trail transfer), log per episode, export to futon3/futon7
+because:Tying simulation to viriya turns FUTON2 into a practice accelerator
+evidence: evidence[Hunger/tau metrics operational] evidence[Colony-level aggregates computable from pivot]

+ NEXT-STEPS:
next[Define viriya indicator formulas] next[Implement colony activation score] next[Export viriya metrics to futon3/futon7] next[Document practice acceleration use cases]

:success-criteria
pass[Viriya indicators computed per episode] pass[Higher futons can read energy health] pass[Metrics correlate with observable colony vigor] fail[Viriya disconnected from actual dynamics]

:psr-example ” Selected viriya metrics for practice integration because simulation should inform real activation”
:pur-template ” Episode {{id}}, viriya score: {{score}}, activation: {{level}}”

! instantiated-by: Prototype 13 — Debugger & Visualiser [😻/由 🚢/一]
:maturity :greenfield
:depends-on [f2/P6]
:evidence-for-active [Clerk notebook plots tau/hunger/error timelines; follow-agent mode traces mu]
+context:Pivot logs are textual and require manual parsing.
+if:You want interactive introspection of mu vectors, tau evolution, and weighted errors
+however:No visualiser or ” follow Alice” mode exists
+then:Build Clerk notebook plotting tau, hunger, errors, pivot timelines; add mode that traces one agent' s full mu timeline
+because:Visual insight accelerates iteration and debugging
+evidence: evidence[Pivot data available for visualization] evidence[trace_tick.clj generates SVG plots]

+ NEXT-STEPS:
next[Build Clerk notebook for AIF visualization]

enables collaboration”
:pur-template ” Training session {{id}}, checks: {{count}}, transcript: {{path}}”

! instantiated-by: Prototype 11 — System Self-Description [🔃/本]
:maturity :greenfield
:depends-on [f3/P1, f1/P0]
:evidence-for-active [Manifest schema defined; stack components queryable in futon1]
+context:Documentation and APIs exist as separate artifacts that humans must correlate.
+if:You want agents to query the system about its own capabilities
+however:No machine-readable manifest makes components first-class entities
+then:Unify documentation and APIs into machine-readable manifest; make stack components first-class entities in futon1; enable agents to query capabilities and self-verify structure
+because:This closes the reflexivity loop—a semantic network that includes a model of itself
+evidence: evidence[futon4/docs/evidence/rc-1.1-evidence.edn seeds machine-readable claim map]

+ NEXT-STEPS:
next[Expand evidence map into manifest schema (components, interfaces, invariants)] next[Ingest manifest into futon1] next[Enable agents to query stack capabilities] next[Add self-verification of running system vs declared structure]

:success-criteria
pass[Manifest schema covers all stack components] pass[Components queryable as futon1 entities] pass[Running system verifiable against manifest] fail[Self-description requires manual correlation]

:psr-example ” Selected self-description for reflexivity because queryable structure enables agent reasoning”
:pur-template ” Manifest query {{component}}, interfaces: {{count}}, verified: {{yes|no}}”

next[Add follow-agent mode for single-ant tracing] next[Plot mu vector evolution over time] next[Integrate with pivot stream for live updates]

`:success-criteria`
pass[Tau/hunger/error plottable interactively] pass[Single agent traceable through full episode] pass[Visualizations update from pivot stream] fail[Debugging requires manual log parsing]

```
:psr-example ” Selected debugger for
development velocity because visual insight
beats log parsing”
:pur-template ” Debug session {{id}}, agent:
{{agent}}, frames: {{count}}”
```