COGNITIVE ATOMS · RESEARCH

Distributed Cognition Infrastructure

A position paper on a research direction for Cognitive Atoms

Ken Ramirez · Cognitive Atoms, Inc. · November 2026 · Version 2 · cognitiveatoms.com/research/distributed-cognition

> **Reader's note.** This is a position paper, not a product description and not a report of experimental results. It asks how persistent, reusable units of cognition could change how AI systems work, and states how that could be tested. Section 1 explains what the current product does and which ideas in this paper remain research.

Abstract

Can some of today's AI work move from repeated, stateless transformer inference to persistent, reusable cognitive structures? This position paper argues that a full replacement of frontier foundation models is not justified, but that a layered architecture is feasible enough to prototype now: models extract, validate, execute, and reuse typed cognitive atoms inside graph-native, signed memory systems that can run locally. The pressure for such a layer is real. Self-attention cost grows quadratically with sequence length, long-context use remains fragile, key-value caches grow linearly with context, and frontier progress is tied to rapidly scaling compute, power, and capital. [1, 2, 3, 4] Agent memory, retrieval, skill libraries, knowledge graphs, and neurosymbolic systems show that parts of persistent, reusable cognition already exist, in fragmented form. [5, 6, 7, 8, 9] We expect such a layer to win first where work is repetitive, auditable, privacy-sensitive, and procedural, and to lose where tasks are novel, poorly typed, or open-ended. This paper presents no new empirical results; it states hypotheses, a cost model, and benchmarks that would test them.

Contributions.

1. A working definition and typed schema for a *cognitive atom*, with a mapping from the cognitive types used in the Cognitive Atoms product today to the formal types (Sections 1 and 4).
2. A cost model for atomized reuse that accounts for retrieval on every task, validation failure, maintenance under drift, and a prompt-caching baseline (Section 6).
3. Five falsifiable hypotheses with metrics, baselines, and pass criteria, and a phased prototype roadmap (Section 7).

> **A note on terms.** "Distributed cognition" is an established idea in cognitive science: cognition is spread across people, artifacts, and time rather than contained in one head [10, 11]. We use the phrase in a narrower, engineering sense: reusable units of cognitive work that persist outside any single model and can move between people, devices, and AI systems. We share that tradition's premise that external structures carry cognition; we do not claim to model human cognition.

1. How this research relates to Cognitive Atoms today

Cognitive Atoms is built on three core capabilities. **Memory** keeps conversation timelines, questions, answers, agent actions, results, and decisions, each with its source and date. **Cognition** connects and interprets that history: what was decided, what changed, which assumptions still hold, and what is still open. **Context** carries it into the next task or AI provider, including ChatGPT, Claude, Claude Code, Gemini, and Grok.

In this paper's terms, the product already works with typed, source-linked units of cognition (decisions, constraints, assumptions, questions, and findings) rather than raw transcripts or summaries. Table 1 in Section 4 maps these product types onto the formal atom schema. That is the first, practical layer of the architecture described below; the executable, signed, and shared layers are not.

Today, a Cognitive Atoms account is hosted: what a user saves is stored on our servers, in their own account, is never sold, and is never used by Cognitive Atoms to train AI models. It is not end-to-end encrypted. When a user carries context into another provider's product, such as ChatGPT, Claude, Gemini, or Grok, that provider's own data policy applies to what it receives. The local-first nodes, signed atom exchange, cognitive intermediate representation, and shared commons in Sections 4 to 7 are research directions. They are not features of the current product.

2. Why centralized transformer systems dominate today

Current frontier AI systems are centralized for structural reasons, not because intelligence is inherently centralized. The original Transformer design makes self-attention attractive because it parallelizes well and connects all positions with short dependency paths, but it also imposes O(n^2 · d) self-attention complexity per layer as sequence length grows. Later work such as FlashAttention improves the memory movement and runtime profile of attention substantially, but it does not remove the underlying long-context pressure that makes bandwidth, cache, and accelerator locality central engineering concerns. Alternative sequence models such as Mamba show that the dominance of transformers is contingent rather than inevitable, but the practical ecosystem today is still built around transformer-compatible hardware and software stacks. [1, 12, 13]

The second reason is memory. Even when autoregressive inference avoids recomputing the entire past sequence, transformer deployments rely on key-value caches whose memory footprint grows linearly with context length. Recent reviews describe KV-cache optimization as a first-order deployment problem because larger context windows directly pressure GPU memory capacity, memory bandwidth, and throughput, especially in production and distributed serving environments. This means that "more context" is not a free substitute for genuine persistent memory; it is often an expensive approximation to persistence. [3]

The third reason is economics of scale. Epoch AI estimates that training compute for frontier language models has grown roughly 5× per year since 2020, and that the largest AI data center, Colossus 2 in Memphis, Tennessee, is about 1.1 million H100-equivalent chips. [4]

Epoch also models a typical one-gigawatt AI data center, owned by a US hyperscaler, at about $38 billion in up-front capital; it presents this as a stylized model rather than an estimate for any specific facility. [14] An analysis of 500 AI supercomputers found that leading AI supercomputer performance doubled every nine months, that the leading system in March 2025 used about 200,000 AI chips and 300 MW of power, and that companies hold a rapidly growing share of total AI supercomputer performance while governments and academia decline. [15] This is why frontier training and serving are drifting toward a small number of firms and sites with capital, interconnects, power procurement, and supply-chain access.

The fourth reason is cost opacity and concentration in model development. Stanford HAI's 2025 AI Index estimated training costs of roughly $79 million for GPT-4 and roughly $170 million for Llama 3.1 405B, while its 2024 edition reported around $191 million for Gemini Ultra; we found no revised dollar estimates in the 2026 edition's research and development summary. The same 2025 report found that the cost of querying a GPT-3.5-level system dropped from $20 per million tokens in late 2022 to $0.07 by October 2024. [16] More recent analysis finds that the cost of LLM inference at a fixed performance milestone has fallen by between 9× and 900× per year, depending on the milestone. [17] Any persistence architecture must beat this moving target; it cannot assume that today's token prices and long-context costs will hold.

Energy adds a fifth constraint. The IEA's 2026 updated projection has global data-center electricity consumption roughly doubling, from about 485 TWh in 2025 to about 950 TWh in 2030, revising its 2025 projection of about 945 TWh. Not all of this growth comes from language models, but it confirms that AI-era compute scaling has grid-level implications. The implication for this paper is narrow: architectures that amortize repeated reasoning, reduce redundant context loading, and shift some cognition to local or edge execution are economically and strategically relevant even if they never replace frontier training. [18]

We also hypothesize that human language is an inefficient substrate for machine-to-machine cognition. It is excellent for social coordination and human interpretability, but it is ambiguous, verbose, and unevenly tokenized. The evidence we cite is narrow: multilingual tokenization research shows that the same content can yield very different token counts across languages, in extreme cases up to 15×, with direct consequences for cost, latency, and usable context. A machine-native cognitive layer should therefore avoid inheriting these token-stream inefficiencies where it can; whether it can do so at acceptable fidelity is an open question. [19]

3. What biology and prior AI systems suggest

Biological intelligence is not a blueprint for digital systems, but it is a useful comparative model. The adult human brain runs on about 20 W and consumes about 20% of the body's energy budget despite being about 2% of body mass. Reviews in neuroscience and NeuroAI repeatedly emphasize that this efficiency is achieved through a combination of sparse activation, locality, dynamical reuse, prediction, shared memory-and-compute substrates, and event-driven signaling rather than through brute-force dense recomputation of all possible pathways at every step. The right lesson is not "silicon should copy cells," but "persistent state and selective activation matter." [20, 21]

Several biological motifs are relevant. Sparse coding work showed that efficient representations of sensory input can emerge by favoring sparse activation patterns. [22] Predictive-coding accounts argue that higher levels of a hierarchy transmit expectations while lower levels surface prediction error, reducing full-state retransmission. [23] Hierarchical-control work argues that behavior relies heavily on reusable abstractions and nested task structure. [24] Chunking research shows that grouping related items improves effective working-memory performance by reusing structure instead of storing every detail independently. [25, 26] None of these findings proves a cognitive-atom architecture, but together they support a research program centered on sparsity, abstraction, reuse, and persistent structured memory.

The energy budget of biological cognition also appears to be dominated by communication more than raw local computation. A cortical energy analysis estimated that communication costs can far exceed computation costs in cortex. That observation parallels modern AI systems, where memory access and movement increasingly bottleneck performance, and where FlashAttention's success came from reducing I/O rather than changing model semantics. For distributed cognition infrastructure, the analogous design principle is straightforward: only synchronize, materialize, and validate what is worth synchronizing, materializing, and validating. [12, 27]

Prior AI work points in the same direction. Knowledge-graph research formalizes graph-based accumulation of entities, relations, schema, identity, and context, [8] and RDF provides a mature open standard for graph interchange and schema evolution. [28] OpenCog's AtomSpace is especially relevant because it treats "atoms" as graph elements that can represent both data and procedures. [29] Among cognitive architectures, ACT-R is explicitly hybrid symbolic and subsymbolic, [30] and Soar combines symbolic working memory with procedural, semantic, and episodic long-term memories. [31] These are not distributed cognition infrastructure, but they show that typed, persistent, reusable cognitive units have a long precedent.

Recent learning systems provide the missing bridge from classic symbolic memory to modern reusable skill formation. RAG combines parametric and non-parametric memory to improve provenance and updateability. DreamCoder learns symbolic abstractions that compress sets of programs into reusable libraries. Stitch and Lilo explicitly optimize library quality by compression and abstraction reuse. Voyager shows that an ever-growing executable skill library can accumulate temporally extended, interpretable, compositional behaviors and transfer them to a new world. MemGPT, A-Mem, and subsequent surveys on agent memory all treat memory as a first-class systems problem rather than a prompt-engineering afterthought. The conceptual leap in this paper is to generalize these ideas from task-specific memory and code reuse to a broader protocol for distributed cognition. [5, 6, 7, 32, 33, 34, 35]

Two recent lines of work are the closest neighbors to the procedure atom proposed here. Buffer of Thoughts stores high-level thought templates distilled from solved problems and retrieves them for new ones, [36] and Agent Workflow Memory induces reusable workflows from agent trajectories and reuses them on later web tasks. [37] ExpeL extracts natural-language insights from an agent's own experience, [38] Generative Agents maintain a memory stream with reflection and retrieval, [39] and GraphRAG, Mem0, and Zep build graph-structured or long-term memory for retrieval and agents. [40, 41, 42] What this paper adds relative to them is not reuse itself but four properties held together: a typed schema that spans declarative

atoms (decisions, constraints, questions) and procedural ones; explicit pre- and postconditions so a reused atom can be validated before it is trusted; provenance links back to source material on every atom; and portability across models and providers rather than reuse inside one agent or one vendor's stack.

4. Cognitive atoms and machine-native cognitive languages

A rigorous working definition is possible. A cognitive atom can be defined as the smallest typed, reusable, validated unit of cognitive work that preserves enough structure to be executed, composed, referenced, and audited outside the original prompt or conversation. It is not "a token," not "a latent vector," and not necessarily "a fact." It is closer to a typed micro-program or typed graph fragment that binds together semantics, preconditions, procedural transforms, provenance, and measured utility. This proposal is inspired by AtomSpace, knowledge graphs, production systems, library learning, and skill libraries, but the exact formalism below is a new synthesis rather than an established standard. [29]

A useful formal schema is:

```
a = ⟨id, type, inputs, pre, body, post, refs, conf, cost, scope, sig⟩

  id     = content-addressed identifier
  type   ∈ { fact, relation, decision, constraint, assumption, question,
             finding, procedure, policy, test, plan, critique, skill }
  inputs = typed variables or graph bindings
  pre    = preconditions over world or model state
  body   = executable transform, rule, or retrieval program
           (empty for purely declarative atoms)
  post   = postconditions or predicted effect
  refs   = provenance links to source atoms, documents, traces, or benchmarks
  conf   = (p, n, t): estimated probability of being correct or successful,
           number of validations behind it, time of last validation
  cost   = (ms, tokens, bytes): estimated latency, model tokens, and memory
  scope  ∈ { private, shared, public }
  sig    = signature, attestation, or validator metadata
```

Table 1. Product types mapped to formal atom types.

Type used in the product today	Formal type	Typical body
Decision	decision	Declarative; refs to the exchange where it was made; may supersede an earlier decision
Constraint	constraint	Declarative; usable as a precondition on plans and procedures
Assumption	assumption	Declarative; unconfirmed until validated or contradicted
Question	question	Declarative; open until a finding answers it
Finding	finding	Declarative; refs to supporting evidence
(not yet in the product)	procedure, policy, test, plan, critique, skill	Executable or evaluable; the research layer of this paper

A cognitive atom should be judged less by linguistic elegance than by reusability, compressive power, execution stability, and auditability. The scope classes match the security posture in Section 6: private atoms never leave the user's trust boundary, shared atoms move among authorized peers, and public atoms enter a commons only after validation. That is the shift from "language output" to "cognitive infrastructure."

A corresponding relation grammar can also be typed and sparse:

```
R  = { causes, enables, contradicts, specializes, abstracts,
       temporally_precedes, retrieves, cites, validates,
       depends_on, composes_with, answers, supersedes }
```

This yields a collective cognition graph $G = (V, E, M)$, where V is a set of cognitive atoms, $E \subseteq V \times V \times R \times [0, 1]$ is a set of typed edges, each with a weight between 0 and 1, and M holds metadata such as namespaces, access control, authorship, and validator sets. Because knowledge-graph research shows that graph models integrate dynamic, heterogeneous knowledge incrementally, a graph is a natural persistent container for atom composition. [8, 43]

A minimal example clarifies the proposal:

```
atom Debug.NullDeref.v1
type: procedure
inputs: {stacktrace, file_graph, test_graph}
pre:
  crash_kind == "null_dereference"
body:
  MATCH source_owner(stacktrace.line) -> candidate_fn
  RETRIEVE prior_fix(pattern="null_guard", scope=project)
  EXEC synthesize_patch(candidate_fn, pattern="guard_or_early_return")
  EXEC run_tests(affected_only=true)
post:
  compile_ok == true
  tests_regressed == 0
refs:
  cites(issue#4821, commit#ab19f)
conf:
  (p=0.82, n=14, t=2026-10-02)
cost:
  (ms=4200, tokens=1800, bytes=2100)
scope:
  private
sig:
  validator = team-ci-key
```

Illustrative example. The values are hypothetical and do not describe a measured result.

This is the level at which "cognition" becomes persistent and reusable: the next similar failure does not require replaying an entire conversation, only retrieving, adapting, and validating typed atoms. The idea is

close to Agent Workflow Memory and Buffer of Thoughts; the difference is the explicit pre- and postconditions and provenance that let the atom be checked before reuse. [7, 36, 37]

The right machine-native language is therefore probably hybrid, not purely symbolic and not purely vectorial. Natural language remains the best human interface. Vector embeddings remain a cheap retrieval substrate for approximate similarity. Symbolic and graph structures remain best for compositionality, auditing, and explicit dependency management, and neurosymbolic graph reasoning exists because neither pure rules nor pure embeddings dominate across the whole problem space. [9] Compiler infrastructure suggests a useful analogy: source languages stay expressive for humans, while optimized systems run through typed intermediate representations such as LLVM IR and MLIR. [44, 45] A plausible future for cognition is analogous: human language at the edges, a cognitive intermediate representation in the middle, and model-specific latent execution beneath it.

A proposed Cognitive Intermediate Representation can therefore be thought of as an MLIR-like “middle layer” for cognitive work:

```
CIR instruction set
LOAD(atom_id)
BIND(var, type)
MATCH(pattern, graph_scope)
RETRIEVE(query, k, mode)
ASSERT(predicate, weight)
EXEC(proc_id, args)
PLAN(goal, constraints)
VERIFY(sig_or_test)
MERGE(graph_delta, policy)
EMIT(atom_or_action)
```

The design goal is not to mimic assembly language but to create a typed, versioned, optimizable cognitive bytecode that can be produced by models, stored in graphs, statically checked, replayed, benchmarked, and partially compiled for local execution. This idea does not yet exist as a mature standard; it is a proposed research object. Its feasibility rests on evidence from compiler IRs, graph query engines, executable hypergraphs, and program-library learning. [29, 32, 45]

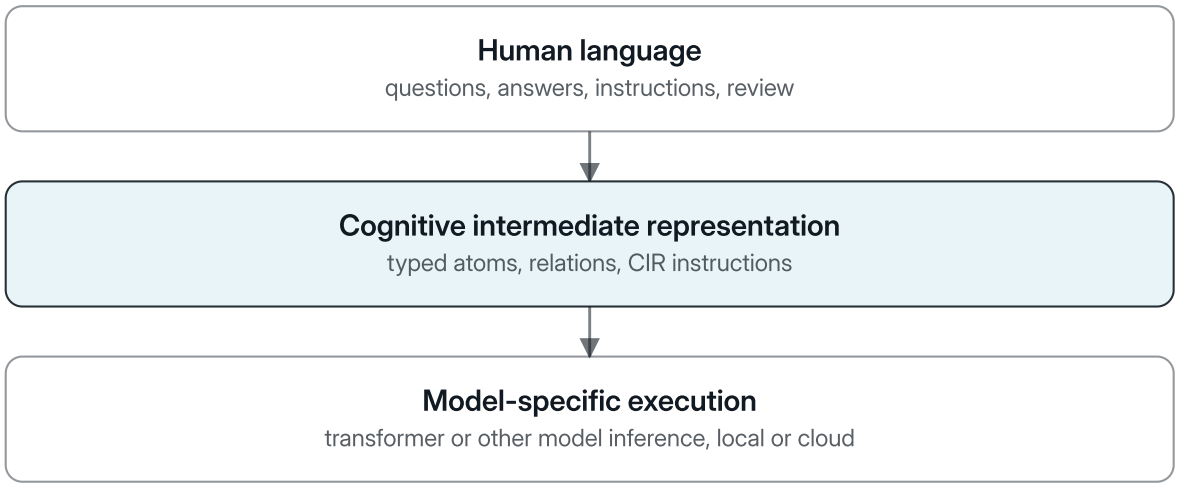

Figure 1. The proposed stack. People work in natural language at the edges; reusable cognition lives in a typed middle layer; models execute beneath it. Only the top layer and part of the middle layer exist in the product today.

5. A research direction: distributed, local-first cognition

> This section describes an architecture we are researching. It is not how the current Cognitive Atoms product stores data; today's accounts are hosted, as described in Section 1.

The most plausible architecture is not one giant cognition service. It is a layered network of local-first cognition nodes connected by optional synchronization and validation protocols. Local-first software research argues that authoritative copies of user data should reside on user devices, with servers acting primarily as sync and backup mechanisms. Federated learning similarly showed that useful learning can occur while raw data stays on client devices. CRDT research provides practical ways to synchronize replicated state while preserving convergence under intermittent connectivity. Combined, these literatures provide a credible substrate for distributed cognition nodes that are offline-capable, user-owned, and incrementally synchronizable. [46, 47, 48]

A cognition node can be decomposed into six services. First, a local memory plane stores private graphs, embeddings, traces, and validated atoms. Second, an extraction plane converts interactions, workflows, code traces, and observations into candidate atoms. Third, a reasoning plane orchestrates one or more local or cloud models for ranking, retrieval, repair, or synthesis. Fourth, a validation plane runs tests, peer checks, or policy constraints and signs accepted atoms. Fifth, a synchronization plane exchanges selected public or federated atoms across peers. Sixth, a governance plane handles namespaces, access control, quotas, and reputation. This is an inference from the component technologies above, not a claim that such a full stack is already solved.

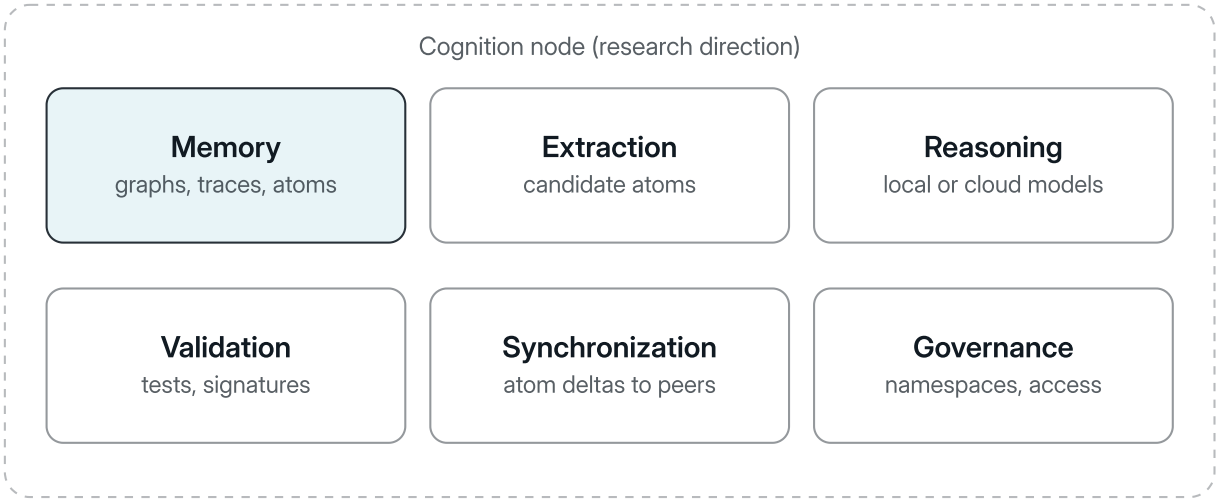

Figure 2. The six planes of a proposed cognition node. Most cognition stays in the local memory plane; only selected, signed atoms leave through the synchronization plane.

The distributed aspect is central. In this model, most cognition accumulates locally, and only reusable, policy-compliant, signed atom subsets propagate. Edge computing research treats devices as active computing nodes rather than passive data collectors, to reduce latency, improve privacy, and avoid wasteful data movement. [49] For cognition infrastructure, the practical implication is that the network should synchronize atom deltas, not full transcripts and not training corpora unless explicitly chosen.

Consensus in such a system should be selective rather than universal. The protocol only needs strong validation for atoms whose reuse value is high or whose failure cost is high. Many atoms can remain private and unshared. Others can be locally signed and later promoted to a public commons after passing benchmark suites, cross-node agreement, or domain-specific tests. This is much closer to package registries, reproducible scientific artifacts, or Git commit workflows than to global blockchain consensus for every transaction. Existing DID standards can provide portable, decentralized identifiers for users, organizations, or artifacts without assuming one central identity provider. [50]

A distributed cognition graph also creates a path for inheritance. A team, city, lab, or ecosystem can accumulate atoms over time, fork them, refine them, and revalidate them. Personal knowledge graph work is especially relevant here: it argues that graph models are well-suited to dynamic, heterogeneous, personally controlled knowledge accumulation. In other words, distributed cognition need not mean "anonymous swarm intelligence." It can mean portable, layered, locally sovereign memory that optionally participates in larger interoperable commons. [43]

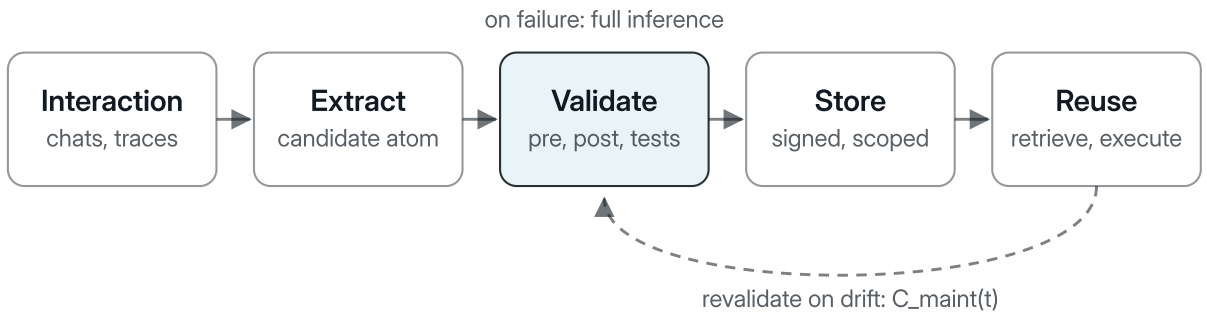

Figure 3. The life of an atom. An atom is trusted only after validation, and is revalidated as code and models drift; a failed check falls back to full inference, which the cost model in Section 6 counts.

6. Economics, governance, security, and autonomy

The economic case for cognitive atoms is not that they make all AI cheap. It is narrower: on recurring workloads they can shift some cost from re-reasoning over a full context to retrieving, adapting, and validating reusable structures. A cost model has to count what reuse actually costs, including retrieval on every task, validation that fails, and upkeep as code and models drift:

```
For a task family T with N tasks over a period t:

  C_baseline = N · C_full

  C_atomized = C_extract + C_store + C_maint(t)
             + N · [ C_retrieve + h·C_validate + h·v·C_exec + (1 − h·v)·C_full ]

  h = hit rate: share of tasks for which a candidate atom is retrieved
  v = validation pass rate, given a hit
  r = h·v = effective reuse rate
  C_maint(t) = cost of revalidating stored atoms as codebases and models drift
```

Retrieval is paid on every task, hit or miss. A retrieved atom that fails validation costs retrieval, validation, and full inference. Atomization is favorable only when r is high, extraction quality is good, and C_retrieve + C_validate is small relative to C_full. The right comparison is cost per solved task at equal quality, not raw cost per task. This is analogous to library induction, where compression and reuse improve future solve efficiency, and to retrieval-augmented generation, where explicit memory can outperform parametric recall in knowledge-intensive settings. [6, 32, 33, 34]

The strongest baseline is not stateless inference but caching. Providers already discount repeated prompt prefixes, and serving systems reuse attention state across requests: Prompt Cache reuses precomputed attention for shared prompt modules, and SGLang's RadixAttention reuses KV cache across calls that share a prefix. A fair C_full therefore assumes caching is on, which narrows the token-cost gap between atomized and cached inference. Caching, however, is tied to one provider, one model, and an exact or near-exact prefix, and it carries no provenance. The advantage atoms can claim over caching is portability across

models and providers, auditability, and reuse across different prompts that share structure, rather than tokens saved. [51, 52]

A compression objective can also be made explicit:

```
Choose an atom basis B to minimize
  L(B, D) = L(B) + L(D | B)
```

where D is a corpus of tasks, traces, or dialogues and L is a description-length cost measured in bytes, graph size, execution cost, or another agreed metric. This is classical Minimum Description Length reasoning adapted to cognition infrastructure. The important point is that "cognitive atoms" should be benchmarked as compression devices and execution devices, not merely as representational curiosities. [53]

Still, the economics are not one-sided. Inference prices are dropping rapidly, model quality is improving, and long-context engineering continues to advance. That means persistent cognition systems will struggle if all they offer is modest token savings. Their stronger economic moat is likely to come from a bundle: reuse, local privacy, faster cold-start performance, auditability, portability across models, and organizational memory retention even when vendors or context windows change. The local-first literature matters here because it frames the value proposition as ownership and durability rather than mere speed. [17, 46]

Governance is not an afterthought, because compute and software control are already concentrating. Ostrom's commons work remains the best starting point for long-lived shared resources: clearly defined boundaries, collective-choice arrangements, monitoring, graduated sanctions, and low-cost conflict resolution all apply to an open cognition commons. [54] More recent work shows that nominal openness in AI does not prevent concentration of power over the ecosystem. [55] The architecture implied here is layered: a nonprofit or public-benefit foundation for the commons, commercial providers at the deployment layer, and private governance at the node layer.

Security and privacy are the hardest operational bottlenecks. Distributed and federated systems are vulnerable to data poisoning, model poisoning, backdoors, and malicious participants. [56] Indirect prompt injection becomes especially serious once tools and agentic workflows are involved, because content an agent reads can act as instructions. [57] Trusted execution environments provide a pragmatic isolation mechanism for sensitive code and data, [58] and decentralized identifiers plus signatures can support provenance and authorship. [50] Zero-knowledge proofs for neural network inference make attestable cognition plausible in principle, but proving overheads and engineering complexity remain significant. [59] Privacy-preserving distributed cognition is possible as a direction, but not yet operationally cheap.

A practical security posture would therefore separate three atom classes. Private atoms never leave the local trust boundary. Shared atoms can move among authorized peers with signatures and encrypted transport. Public atoms enter the commons only after benchmarked validation and provenance checks. This split does not eliminate attacks, but it reduces the blast radius of poisoning and makes "cognition ownership" implementable rather than rhetorical.

7. Feasibility, falsifiable hypotheses, and prototype roadmap

The technically feasible part today is substantial. One can build a local cognition node using a graph store, vector index, small local models or cloud APIs, extraction pipelines, signed artifacts, CRDT-style sync, and a test-backed validation layer. The pieces already exist in adjacent form: knowledge graphs, RDF, agent-memory systems, local-first sync, federated training methods, edge AI, and skill libraries. What does not yet exist is a generally reliable compiler from messy human interaction into compact, transferable, executable atoms with low enough error to justify wide reuse. That is the core research gap. [8, 43]

Table 2 states each claim as a hypothesis with a metric, a baseline, and a pass criterion. The thresholds are our proposals; they make the claims falsifiable and can be tightened by domain.

Table 2. Falsifiable hypotheses.

	Hypothesis	Metric	Baseline	Pass criterion
H1	Atomization lowers cost on recurring task families	Cost per solved task, and task success rate	Same model with provider prompt caching on	At least 30% lower cost per solved task, with no more than 2 points lower success
H2	Atomized memory improves long-horizon recall	LongMemEval and LoCoMo scores	Full-context prompting and a retrieval-only memory	Higher score than both at equal or lower context tokens
H3	Typed, source-linked atoms improve auditability	Share of answers whose claims trace to a correct source; time for a reviewer to verify a claim	Summary-based memory without source links	At least 90% traceable, and verification at least 2× faster
H4	Local-first nodes reduce privacy exposure	Bytes of user content leaving the device per task; leakage under a red-team protocol	Hosted memory of the same content	At least 10× fewer bytes leaving the device, and no higher leakage
H5	Selective sync of signed atom deltas converges	Time to convergence and conflict rate under partitioned, offline edits	Last-writer-wins sync of full records	Converges in all test runs, with fewer unresolved conflicts than the baseline

A minimal benchmark suite should include: atom extraction precision and recall against expert-annotated workflows; reuse hit rate h and validation pass rate v on held-out tasks; compression ratio measured in bytes, from raw transcript to serialized atom graph; quality retention under atomized execution versus baseline prompting; dollars, joules, and latency per solved task; convergence under disconnected sync; poisoning resilience under adversarial atom injection; and privacy leakage under different sharing scopes. LongMemEval and LoCoMo provide useful templates because they test long-range temporal and causal recall rather than short-context imitation. [60, 61]

A realistic roadmap has four phases. Phase one is a single-node MVP: local graph store, embedding index, atom schema, and a narrow extractor/validator for one repetitive domain such as debugging, document triage, or architecture review. Phase two adds a cognitive IR and execution engine: atoms become replayable, testable, and rankable rather than mere records. Phase three adds selective sync: signed atom exchange, namespace control, conflict handling, and local-first replication between devices or trusted teams. Phase four adds federated validation and governance: shared benchmarks, reputation, portable

identity, and a public commons for high-value atoms that survive repeated validation. Each phase is feasible with existing ingredients, but each phase also creates new failure modes that must be measured rather than assumed away. [46]

The near-term judgment is therefore mixed. Immediate prototypes are feasible now. Medium-term infrastructure is plausible if atom extraction, compression, and validation improve materially. Long-term speculative possibilities such as a global open cognition commons with machine-native compressed reasoning languages are conceivable but far from established. The strongest version of the thesis is not that foundation models disappear. It is that intelligence may increasingly look like a stack: large models as general-purpose priors, cognitive-IR layers as reusable reasoning infrastructure, and local signed graphs as durable memory and sovereignty boundaries.

8. Open questions and limitations

The largest uncertainty is atom induction fidelity. Current models are still brittle in long-context use, still challenged by long-range temporal and causal coherence, and still vulnerable to memory contamination and prompt-level attacks. If extraction quality is poor, atom systems could simply turn transient errors into persistent errors. [2, 57, 61]

A second uncertainty is representational universality. It is not yet clear whether one cognitive IR can span facts, procedures, plans, critiques, causal models, and social coordination with acceptable complexity. Existing systems split across graphs, embeddings, symbolic rules, code, and prompts for a reason: each substrate has different strengths. The most defensible current position is hybridism, not a claim that one perfect machine language has already been found. [9]

A third limitation is economics under fast-moving hardware and model markets. Falling inference prices, better caching, more efficient sequence models, improved sparsity, and edge-model optimization all reduce the margin by which atom systems can win on pure cost. If cognitive-atom systems succeed, it will likely be because they combine cost advantages with persistence, auditability, sovereignty, and reuse in domains where those properties matter enough to pay for the added complexity. [16, 17, 51]

The final limitation is governance. A distributed cognition protocol that ignores capture, poisoning, identity, and stewardship will recreate the same centralization problems in a new vocabulary. That is why the governance architecture must be treated as part of the system design, not as a post hoc policy layer. [54, 55]

Finally, this is a position paper. It reports no new measurements, and every claim in it should be read as a hypothesis until tested against the benchmarks in Section 7.

Competing interests

The author is the founder and chief executive of Cognitive Atoms, Inc., which builds a product motivated by the ideas in this paper. The company funded this work. No other funding was received.

References

1. Vaswani et al. (2017). Attention Is All You Need. NeurIPS 2017. arxiv.org/abs/1706.03762
2. Liu et al. (2024). Lost in the Middle: How Language Models Use Long Contexts. TACL 12. aclanthology.org/2024.tacl-1.9/
3. Xu, Khaira and Singh (2026). KV Cache Optimization Strategies for Scalable and Efficient LLM Inference. arXiv:2603.20397. arxiv.org/abs/2603.20397
4. Epoch AI. Trends in AI (updated 5 February 2026; accessed 11 November 2026). epoch.ai/trends
5. Packer et al. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560. arxiv.org/abs/2310.08560
6. Lewis et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020. arxiv.org/abs/2005.11401
7. Wang et al. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291. arxiv.org/abs/2305.16291
8. Hogan et al. (2021). Knowledge Graphs. ACM Computing Surveys 54(4). arxiv.org/abs/2003.02320
9. DeLong, Fernández Mir and Fleuriot (2023). Neurosymbolic AI for Reasoning over Knowledge Graphs: A Survey. arXiv:2302.07200. arxiv.org/abs/2302.07200
10. Hutchins (1995). Cognition in the Wild. MIT Press. mitpress.mit.edu/9780262581462/cognition-in-the-wild/
11. Hollan, Hutchins and Kirsh (2000). Distributed cognition: toward a new foundation for human-computer interaction research. ACM TOCHI 7(2). doi.org/10.1145/353485.353487
12. Dao, Fu, Ermon, Rudra and Ré (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. NeurIPS 2022. arxiv.org/abs/2205.14135
13. Gu and Dao (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. COLM 2024. arxiv.org/abs/2312.00752
14. Michael and Cottier (2026). Servers account for 60% of the total cost of ownership of a one-gigawatt AI data center. Epoch AI data insight, 14 May 2026 (accessed 11 November 2026). epoch.ai/data-insights/ai-datacenter-cost-breakdown
15. Pilz, Sanders, Rahman and Heim (2025). Trends in AI Supercomputers. arXiv:2504.16026. arxiv.org/abs/2504.16026
16. Stanford HAI (2025). AI Index Report 2025 (accessed 11 November 2026). hai.stanford.edu/ai-index/2025-ai-index-report
17. Epoch AI. Key research findings: cost of LLM inference at a fixed performance milestone (accessed 11 November 2026). epoch.ai/llms.txt
18. International Energy Agency (2026). Key Questions on Energy and AI: updated projections, executive summary (accessed 11 November 2026). www.iea.org/reports/key-questions-on-energy-and-ai/executive-summary
19. Petrov, La Malfa, Torr and Bibi (2023). Language Model Tokenizers Introduce Unfairness Between Languages. NeurIPS 2023. arxiv.org/abs/2305.15425
20. Raichle and Gusnard (2002). Appraising the brain's energy budget. PNAS 99(16). pmc.ncbi.nlm.nih.gov/articles/PMC124895/
21. Zador et al. (2023). Catalyzing next-generation Artificial Intelligence through NeuroAI. Nature Communications 14. doi.org/10.1038/s41467-023-37180-x
22. Olshausen and Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381. www.nature.com/articles/381607a0
23. Rao and Ballard (1999). Predictive coding in the visual cortex. Nature Neuroscience 2. doi.org/10.1038/4580
24. Botvinick (2008). Hierarchical models of behavior and prefrontal function. Trends in Cognitive Sciences 12(5). doi.org/10.1016/j.tics.2008.02.009
25. Miller (1956). The magical number seven, plus or minus two. Psychological Review 63(2). doi.org/10.1037/h0043158
26. Thalmann, Souza and Oberauer (2019). How does chunking help working memory? Journal of Experimental Psychology: Learning, Memory, and Cognition 45(1). doi.org/10.1037/xlm0000578

27. Levy and Calvert (2021). Communication consumes 35 times more energy than computation in the human cortex. PNAS 118(18). pmc.ncbi.nlm.nih.gov/articles/PMC8106317/
28. W3C (2014). RDF 1.1 Concepts and Abstract Syntax (accessed 11 November 2026). www.w3.org/TR/rdf11-concepts/
29. OpenCog. AtomSpace (accessed 11 November 2026). wiki.opencog.org/w/AtomSpace
30. Anderson et al. (2004). An integrated theory of the mind. Psychological Review 111(4). doi.org/10.1037/0033-295X.111.4.1036
31. Laird (2022). Introduction to the Soar Cognitive Architecture. arXiv:2205.03854. arxiv.org/abs/2205.03854
32. Ellis et al. (2021). DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. PLDI 2021. arxiv.org/abs/2006.08381
33. Bowers et al. (2023). Top-Down Synthesis for Library Learning. POPL 2023. arxiv.org/abs/2211.16605
34. Grand et al. (2024). LILO: Learning Interpretable Libraries by Compressing and Documenting Code. ICLR 2024. arxiv.org/abs/2310.19791
35. Xu et al. (2025). A-MEM: Agentic Memory for LLM Agents. arXiv:2502.12110. arxiv.org/abs/2502.12110
36. Yang et al. (2024). Buffer of Thoughts: Thought-Augmented Reasoning with Large Language Models. NeurIPS 2024. arxiv.org/abs/2406.04271
37. Wang, Mao, Fried and Neubig (2024). Agent Workflow Memory. arXiv:2409.07429. arxiv.org/abs/2409.07429
38. Zhao et al. (2024). ExpeL: LLM Agents Are Experiential Learners. AAAI 2024. arxiv.org/abs/2308.10144
39. Park et al. (2023). Generative Agents: Interactive Simulacra of Human Behavior. UIST 2023. arxiv.org/abs/2304.03442
40. Edge et al. (2024). From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130. arxiv.org/abs/2404.16130
41. Chhikara et al. (2025). Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. arXiv:2504.19413. arxiv.org/abs/2504.19413
42. Rasmussen et al. (2025). Zep: A Temporal Knowledge Graph Architecture for Agent Memory. arXiv:2501.13956. arxiv.org/abs/2501.13956
43. Skjæveland et al. (2023). An Ecosystem for Personal Knowledge Graphs: A Survey and Research Roadmap. arXiv:2304.09572. arxiv.org/abs/2304.09572
44. Lattner and Adve (2004). LLVM: A Compilation Framework for Lifelong Program Analysis and Transformation. CGO 2004. doi.org/10.1109/CGO.2004.1281665
45. Lattner et al. (2021). MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. CGO 2021. arxiv.org/abs/2002.11054
46. Kleppmann, Wiggins, van Hardenberg and McGranaghan (2019). Local-first software: you own your data, in spite of the cloud. Onward! 2019. www.inkandswitch.com/essay/local-first/
47. McMahan et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. AISTATS 2017. arxiv.org/abs/1602.05629
48. Shapiro, Preguiça, Baquero and Zawirski (2011). Conflict-free Replicated Data Types. SSS 2011. inria.hal.science/inria-00609399
49. Shi, Cao, Zhang, Li and Xu (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal 3(5). doi.org/10.1109/JIOT.2016.2579198
50. W3C. Decentralized Identifiers (DIDs) v1.1 (accessed 11 November 2026). www.w3.org/TR/did-1.1/
51. Gim et al. (2024). Prompt Cache: Modular Attention Reuse for Low-Latency Inference. MLSys 2024. arxiv.org/abs/2311.04934
52. Zheng et al. (2024). SGLang: Efficient Execution of Structured Language Model Programs. NeurIPS 2024. arxiv.org/abs/2312.07104
53. Rissanen (1978). Modeling by shortest data description. Automatica 14(5). doi.org/10.1016/0005-1098(78)90005-5

54. Ostrom (1990). Governing the Commons. Cambridge University Press. doi.org/10.1017/CBO9780511807763
55. Widder, Whittaker and West (2024). Why 'open' AI systems are actually closed, and why this matters. Nature 635. doi.org/10.1038/s41586-024-08141-1
56. Li and Abdelmoniem (2025). Mitigating Malicious Attacks in Federated Learning via Confidence-aware Defense. arXiv:2408.02813. arxiv.org/abs/2408.02813
57. Greshake et al. (2023). Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. AISec 2023. arxiv.org/abs/2302.12173
58. Sabt, Achemlal and Bouabdallah (2015). Trusted Execution Environment: What It Is, and What It Is Not. IEEE TrustCom 2015. doi.org/10.1109/Trustcom.2015.357
59. Kang, Hashimoto, Stoica and Sun (2022). Scaling up Trustless DNN Inference with Zero-Knowledge Proofs. arXiv:2210.08674. arxiv.org/abs/2210.08674
60. Wu et al. (2025). LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. ICLR 2025. arxiv.org/abs/2410.10813
61. Maharana et al. (2024). Evaluating Very Long-Term Conversational Memory of LLM Agents. ACL 2024. aclanthology.org/2024.acl-long.747/

Version 2, November 2026. The first draft (June 2026) cited earlier figures from Epoch AI and the IEA, which are updated here.