

The Missing Substrate

Why the Era of Experience Needs a Protocol for Knowledge, Not Just Compute

Keif Gwinn — February 2026

Web of Thought Protocol · [wot.ocks](#) · [wot.technology](#) · [now.pub](#)

1. The Second Bitter Lesson

In 2019, Richard Sutton published *The Bitter Lesson*, which has become the most cited essay in modern AI. Its first lesson is well understood: general methods that leverage computation beat human-engineered knowledge. Always. LLMs proved this comprehensively. Scale wins. More data. More parameters. The field absorbed this lesson and built accordingly.

But the essay contains a second lesson that the field has largely ignored:

We should build in only the meta-methods that can find and capture this arbitrary complexity. We want AI agents that can discover like we can, not which contain what we have discovered.

LLMs contain. They don't discover. A model trained on a static dataset, aligned through RLHF, and deployed with frozen weights will process a billion conversations and learn nothing from any of them. Test-time compute (o1, o3, R1) improves answers by searching harder through frozen representations — but the model itself never improves from use. It is, as John Carmack put it, an "IID blender": a sophisticated compression of everything humanity has written, capable of recombining that knowledge but incapable of generating new knowledge through experience.

Sutton is now at Keen Technologies with Carmack, building physical robots that learn Atari through actual joystick gameplay — real latency, real noise, real sparse rewards. No simulation. No imitation learning. Two Turing Award-calibre researchers rejecting the dominant paradigm. This is not a minor disagreement about architecture. It is a claim that the current trajectory — scaling transformers on static data — is a local maximum that will be displaced by systems that actually learn from experience at scale.

The Sequoia Capital analysis of Sutton's position frames this as the transition to the "Era of Experience": AI systems that move beyond training on human-generated text and images to learning directly from their interactions with the world. The Alberta Plan, Sutton's 12-step research roadmap, advocates for "algorithms for acquiring and organizing knowledge rather than encoding knowledge directly." Continual learning. Meta-learning. Systems that improve over their entire operational lifetime, not just during a training phase.

This paper does not argue for or against Sutton's prediction about where intelligence research leads. It argues something more immediate: whatever architecture eventually learns from experience at scale — whether that's reinforcement learning, hybrid neuro-symbolic systems, or something not yet invented — it will need a substrate for storing, attributing, branching, and retrieving experience with full provenance. That substrate does not exist today. The Web of Thought protocol (WoT) is designed to be it.

2. The Infrastructure Gap

Consider what the current AI pipeline looks like:

Internet → scrape → static dataset → train → RLHF → freeze → deploy

At every stage, information is lost:

No provenance. Training data is scraped without attribution. A model's output cannot be traced back to the specific training examples that informed it. When a model confidently states a falsehood, there is no chain of evidence to walk backward and understand *why* it believes what it believes. The model's internal state is a compressed statistical summary — useful, but opaque.

No branching. Every fine-tune, every RLHF run, every alignment intervention is a fork. But the relationship between forks is not preserved. There is no way to compare what a model knew before and after a training intervention, no way to understand which experiences led to which capabilities, no way to A/B test learning strategies at the data layer.

No attribution. The humans who created the training data — writers, coders, artists, researchers — receive no credit and have no visibility into how their work was used. The model's capabilities emerge from millions of human contributions, but those contributions are laundered through tokenisation into an undifferentiated statistical mass.

No trust signal. Not all training data is equal. Some sources are authoritative, some are wrong, some are adversarial. Current pipelines handle this through crude filtering (blocklists, quality scores, deduplication) rather than through structured trust computation. A Wikipedia citation and a Reddit shitpost receive the same treatment once tokenised.

Experience is discarded. Every conversation with a deployed model generates experience data. User corrections, clarifications, task completions, and failures are all signals that could improve future systems. In practice, most of this is discarded or locked in proprietary stores with no standard format, no interoperability, and no way for the user to own their contribution.

This is not a theoretical concern. OpenAI's recently published work on their internal data agent — operating across 600 petabytes and 70,000 datasets — demonstrates the practical consequences. Their system requires "multilayered contextual grounding" (table usage metadata, human annotations, code-extracted definitions) to produce trustworthy answers. They built a "golden SQL" evaluation system to catch output drift. They implemented reusable analytical workflows and conversational memory to avoid re-deriving the same knowledge. Every one of these features is a bespoke, proprietary solution to a problem that could be solved at the protocol level — if a protocol existed.

The Hacker News discussion of OpenAI's work converged on the same insight from multiple directions. Veezoo, after 10 years building natural language analytics, concluded that text-to-SQL on a probabilistic foundation cannot satisfy a CFO who needs numbers that are correct 100% of the time — they built a Knowledge Graph abstraction layer where AI translates natural language to semantic queries that compile deterministically. Another commenter described building "canonical metrics" that are human-curated, with the agent becoming a router from user intent to verified definitions rather than a SQL generator. Multiple

participants agreed: data problems are organisational problems, not technology problems. Conway's Law all the way down.

All of these teams are independently rebuilding partial, proprietary versions of the same infrastructure: a graph of knowledge with provenance, attribution, trust, and human-curated canonical definitions. Each implementation is siloed. None interoperate. The wheel is being reinvented inside every organisation that reaches sufficient data complexity.

3. The Bitter Lesson Applied to Data Infrastructure

Rahman's Threads analysis of Sutton's essay makes a sharp observation about current agent architectures: every rigid workflow you design — planner routes to researcher routes to writer routes to critic — encodes your assumptions about how problems should decompose. That's the expert system move. The Bitter Lesson says that bet loses. Every time. Given enough compute.

The same principle applies to data infrastructure.

Every semantic layer, every metric store, every knowledge graph, every RAG pipeline is an opinionated encoding of how knowledge should be structured. These systems work — just as expert chess systems worked, just as hand-crafted NLP features worked — until general methods with sufficient compute eat the field.

The lesson is not that structure is useless. It's that the *right* structure is minimal, general, and designed to scale with compute rather than against it. The right structure doesn't encode domain knowledge. It encodes meta-methods: the ability to discover, attribute, branch, and trust — regardless of what's being discovered, attributed, branched, or trusted.

HTTP doesn't know what a webpage is about. TCP doesn't know what data it carries. DNS doesn't know what a domain is for. These protocols succeed because they provide minimal, general infrastructure that every application builds on top of. They encode meta-methods for addressing, routing, and resolution — not knowledge of any specific domain.

The AI field needs the equivalent for experience and knowledge: a protocol that doesn't know what an experience is about, but knows how to store it immutably, attribute it cryptographically, connect it to other experiences, weight it by trust, and trace its provenance.

4. Web of Thought: One Primitive

The Web of Thought protocol (WoT) is built on a single primitive: the **Thought**.

A Thought is an immutable, content-addressed, cryptographically signed unit of knowledge:

rust

```

struct Thought {
  cid: Cid,           // BLAKE3 hash of canonical content
  r#type: String,     // Schema type identifier
  content: Value,     // Type-specific payload
  created_by: Cid,    // Identity CID (required, never null)
  created_at: i64,    // Unix timestamp
  source: Option<Cid>, // Input attribution
  because: Vec<ContentRef>, // Provenance chain
  signature: Signature, // Ed25519 over canonical form
}

```

Everything else — connections, attestations, identities, schemas, pools — is a Thought with typed content. There are no separate primitives for relationships, trust, governance, or access control. These emerge from the same structure through different content types and connection patterns.

This is a deliberate design choice aligned with the Bitter Lesson. WoT does not encode domain knowledge. It does not prescribe how learning should work, how knowledge should be organised, or how agents should reason. It provides five capabilities, all derived from one primitive:

Attribution. Every thought has a `created_by` field pointing to a cryptographic identity. Every thought is signed. Authorship is not metadata — it is cryptographic proof.

Provenance. The `because` field is a vector of content references linking to the thoughts that caused this thought to exist. Walk the chain backward and you get full lineage: this conclusion exists *because* of that evidence, which exists *because* of that observation, which was created by that identity at that time.

Trust. Attestations are thoughts about other thoughts, carrying a weight from -1.0 to +1.0 and signed by the attesting identity. Trust is computed from the graph (vouch chains, attestation weights, provenance depth) — never stored as a static property. The same thought has different trust scores for different observers in different contexts.

Branching. Pools are governance-bounded sync boundaries. Fork a pool and you have two branches of the same knowledge graph with different trust weights, different membership, different attestation patterns. Same experiences, different evaluation — the data-layer equivalent of different training runs.

Immutability with correction. Thoughts never change. Corrections are new thoughts with `rework` connections to what they correct. The wrong answer is preserved. The correction is attested. The reason for the correction lives in the `because` chain. Full edit history for any content type — what `git blame` provides for code, WoT provides for all media, all knowledge, all experience.

5. Experience as Graph

What does "learning from experience" look like at the data layer?

An agent interacts with the world. It observes, acts, receives feedback, adjusts. Each of these is a discrete event with attribution, provenance, and — crucially — evaluation by the entities that care about the outcome.

In WoT terms:

The observation is a Thought, created by the agent's perception system, with `source` pointing to the input modality (camera, microphone, API, sensor) and `because` linking to the context that prompted the observation.

The action is a Thought with `because` pointing to the observation and the reasoning that led to the decision. This is the agent's equivalent of "I did X because I saw Y and my policy predicted Z."

The reward signal is an Attestation on the action thought, signed by whatever entity evaluates performance — a human supervisor, an automated reward function, a peer agent, or the environment itself. The weight carries the magnitude. The `because` chain on the attestation carries the evaluation methodology.

The correction is a Rework chain: the original action preserved, the correction attested by the correcting entity, the reason for correction grounded in the `because` chain.

The learning is whatever the learning architecture does with this graph. WoT doesn't prescribe it. The graph is the memory. The learning algorithm reads the graph, updates its representations, and produces new thoughts (predictions, policies, value estimates) that are themselves attributed, signed, and provenance-tracked.

This maps directly to the Alberta Plan's base agent architecture: perception, policy, value function, and transition model — each producing outputs that are thoughts in the graph, each grounded in the experiences that informed them, each traceable back to source.

What this enables that current infrastructure does not:

Trace capability to experience. When an agent demonstrates a novel capability, walk the `because` chain backward: which experiences, which reward signals, which corrections led to this? Not a statistical attribution across billions of parameters, but a concrete, auditable trail through the experience graph.

Branch learning strategies. Fork a pool. Same experience graph. Different attestation weights (different reward functions). Different trust computation (different evaluation criteria). Run two learning strategies on the same data and compare outcomes — at the protocol level, not inside a monolithic training pipeline.

Preserve dead ends. Sutton's whole point is that approaches that look like dead ends today become breakthroughs when compute catches up. In WoT, failed experiments aren't deleted. They're thoughts with low-trust attestations. The experience is preserved. The evaluation is timestamped. When a new learning architecture needs to re-evaluate old experiences with new methods, the data is there — with full context about why it was originally judged as a failure.

Federate experience across agents. Multiple agents learning in different environments can share experiences through pool membership without sharing internal representations. The experiences are CID-addressed and signed — an agent can verify the provenance and trust of experiences from other agents before incorporating them into its own learning. This is continual learning with provenance across organisational boundaries.

6. What WoT Is Not

WoT is not an AI system. It doesn't learn. It doesn't reason. It doesn't generate. It stores, attributes, connects, and enables trust computation on the results of systems that do these things.

WoT is not competing with LLMs. Current LLMs are useful and will remain so. WoT provides the substrate on which LLM outputs — and the outputs of whatever comes after LLMs — can be stored with provenance, attributed to their source, and evaluated by the entities that consume them.

WoT is not a training pipeline. It doesn't prescribe how to use experience data for training. It provides the data in a format that any training pipeline can consume, with provenance and trust signals intact.

WoT does not encode domain knowledge. Schemas are self-describing type definitions — thoughts about the shape of other thoughts. They are created by users, not by the protocol. The protocol doesn't know what "revenue" means or how "medical triage" works. It knows that a thought exists, was created by an identity, has provenance, and has been attested to by other identities with computable trust.

This is Sutton's second lesson applied to infrastructure: build the meta-methods, not the methods. Build the structure that enables discovery, not the structure that contains discoveries.

7. The Model Provenance Problem

A concrete example illustrates why this matters now, not in some future Era of Experience.

When a model claims to be "Claude Opus 4.5" or "GPT-5," there is no cryptographic proof. The model's identity is asserted by the system prompt — text that can be changed, spoofed, or misconfigured. During the development of the WoT specification, a model routing failure demonstrated this precisely: the model changed mid-conversation without any verifiable signal to the user. The protocol being designed to solve trust was being built over a channel that couldn't prove its own identity.

In WoT:

Response thought

created_by: cid:blake3:model_identity

source: connection → uses_input → model/opus-4.5

signature: ed25519 signed by model key

Provider attests model identity via vouch chain.

Consumer verifies signature against known identity.

Mismatch = caught. Auditable.

Every AI response becomes a signed, CID-addressed thought with verifiable provenance to the model that produced it. The provider attests the model identity. The consumer verifies. Model swaps, capability changes, and version drift become detectable — not through trust in the provider's claims, but through cryptographic verification.

This is not a theoretical concern. As AI outputs increasingly inform business decisions, legal proceedings, medical assessments, and public discourse, the inability to verify *which model produced which output* is a growing liability. WoT provides model provenance as a structural property of every output, not as an afterthought bolted onto a system that was designed without it.

8. The Evaluation Pattern

OpenAI's data agent uses "golden SQL" evaluation: known-correct query results against which agent outputs are continuously tested. This is a good practice. In WoT, it requires no new infrastructure.

Expectation thought: "Revenue for Q3 should be £X"

type: schema/expectation

attested by: CFO identity (weight 1.0)

because: [canonical_metric_definition]

Agent answer: "Revenue for Q3 is £Y"

type: schema/answer

because: [query, dataset]

Connection: answer → responds_to → expectation

Automated attestation on the connection:

weight: match(X, Y) ? 1.0 : -1.0

because: [comparison_methodology]

The expectation is a thought. The answer is a thought. The test is a connection. The pass/fail is an attestation on the connection. The methodology is a **because** chain.

No evaluation framework. No separate infrastructure. The same primitives that store knowledge also evaluate it. The same trust computation that weights human opinions also weights automated checks. The evaluation becomes part of the knowledge graph — traceable, auditable, and itself subject to attestation.

This extends naturally to the continual evaluation that experiential learning requires: every reward signal is an attestation, every performance benchmark is an expectation thought, every regression is a detectable drop in attestation confidence on a connection between outputs and expectations.

9. Organisational Knowledge as Protocol

The recurring observation from practitioners building data agents — "data problems are organisational problems, not technology problems" — maps directly to WoT's governance model.

Pools are sync boundaries with governance. Pool governance models (solo, bilateral, threshold, delegated) map to organisational decision structures. When finance and marketing both define "revenue" differently, both definitions exist as thoughts in their respective pools, attested by their respective authorities. There is no conflict to resolve at the protocol level — there are two thoughts with different trust scores depending on which pool (which organisational context) you're querying from.

The "duelling dashboards" problem — multiple teams producing contradictory metrics from the same data — is an attestation problem. Both dashboards are grounded in data. Both are attested by their creators. The question is: whose attestation carries more weight in this context? That's trust computation, and it produces a different answer depending on the observer's position in the organisational graph.

WoT doesn't solve organisational dysfunction. But it makes organisational knowledge structures *visible and computable* rather than implicit and contested. The same metric definition, attested by the CFO in the finance pool, carries different trust weight than the same definition attested by a junior analyst in a personal pool. The graph makes the authority structure explicit without encoding it as a rigid hierarchy.

10. The Neutral Layer

The AI field is fractured by methodological wars with no shared ground for resolution.

In autonomous driving: Waymo bets on expensive real-world data collection; Tesla bets on vision-only fleet learning from billions of miles driven; a growing simulation-first camp bets on synthetic data — cheaper, faster, infinite edge cases, questionable transfer to the physical world. Each camp argues their data is better. The arguments are conducted through competing papers, competing benchmarks, and competing press releases. There is no shared framework to evaluate the claims against each other on common terms.

In language modelling: scaling advocates argue for more parameters and more data; the reinforcement learning camp argues that static training on human text is a dead end; hybrid approaches argue for combinations. RLHF competes with constitutional AI competes with debate-based alignment. Fine-tuning competes with RAG competes with long-context retrieval. Each methodology produces results. None can be compared against the others on a shared substrate with common provenance.

In training data itself: synthetic data proponents argue that LLM-generated training examples can bootstrap capabilities cheaper than human-curated data; sceptics argue that model collapse is inevitable when models train on their own outputs. Both sides are probably right in different contexts. But "different contexts" is precisely what neither side can specify rigorously — because there is no shared layer that tracks which data came from where, how it was validated, and what trust it has earned through accumulated evidence.

WoT does not pick a side in any of these debates. It provides the structural layer on which competing approaches can be compared.

Every data source — real-world sensor, simulator, LLM-generated synthetic, human-curated — is a thought with a source field identifying its origin and a because chain grounding its provenance. Every evaluation is an attestation with a weight, signed by the evaluating entity, traceable to methodology. Every validation — a synthetic prediction confirmed by real-world observation, a fine-tuned model output matching a human expert's judgement — is a connection between thoughts with attestations that accumulate into trust scores.

Over thousands of these validation pairs, the trust profile of each source emerges from the graph. The simulator is reliable for urban intersections (many validated matches), unreliable for adverse weather (divergent results), and untested for construction zones (no validation data). The synthetic training data improves reasoning benchmarks (attested by eval suite) but degrades factual accuracy (attested by human reviewers). The RLHF-aligned model outperforms the constitutional AI model on helpfulness (attested by users) but underperforms on refusal accuracy (attested by red team).

None of these evaluations require anyone to win the argument. They require accumulated evidence in a shared graph with verifiable provenance. The argument resolves itself — not through advocacy, but through receipts.

This is the deeper point about infrastructure. HTTP didn't pick a side between static pages and dynamic applications. TCP didn't pick a side between file transfer and streaming. These protocols succeeded because they provided a neutral layer general enough to support every application built on top of them, including applications their designers never imagined.

WoT is that layer for knowledge and experience. Not because it's impartial — because it's structural. It doesn't have an opinion about which approach to AI works. It has a graph that tracks which approach produced which results, attested by whom, validated against what, in which context, with what provenance. The competing methodologies stop being religious positions and start being branches in a shared experience graph — forkable, comparable, and evaluated by the same trust computation.

Sutton's second lesson says: build the meta-methods, not the methods.

WoT is the meta-method for knowledge infrastructure. The Era of Experience needs it. Current AI systems need it. And every methodological debate in the field needs it — not to resolve the debate, but to give it a substrate where evidence accumulates structurally rather than rhetorically.

The protocol exists. The specification is open. The primitives are proven. What's missing is the recognition that this is infrastructure, not application — and that infrastructure precedes the applications it enables, just as HTTP preceded the web applications that nobody could have predicted when the protocol was designed.

References

Sutton, R. S. (2019). *The Bitter Lesson*. incompleteideas.net/IncIdeas/BitterLesson.html

Sutton, R. S., Bowling, M. H., & Pilarski, P. M. (2022). *The Alberta Plan for AI Research*. [arXiv:2208.11173](https://arxiv.org/abs/2208.11173)

Sequoia Capital (2025). *Richard Sutton's Second Bitter Lesson*. Inference by Sequoia Capital, September 2025.

Rahman, S. (2026). *Thread on Sutton's Bitter Lesson and the Era of Experience*. Threads, February 2026.

OpenAI (2026). *Inside Our In-House Data Agent*. openai.com/index/inside-our-in-house-data-agent/

Gwinn, K. (2026). *Web of Thought Protocol Specification v0.10*. wot.rock.solutions

Appendix: WoT Primitive Summary

Concept	Implementation	Implication
Thought	CID-addressed, signed, immutable	Experience is permanent
Connection	Typed relation between CIDs	Knowledge has structure
Attestation	Weighted opinion on any thought	Trust is computable
Because	Content references to provenance	Lineage is traceable
Rework	Connection preserving edit history	Corrections don't destroy

Concept	Implementation	Implication
Pool	Governance-bounded sync boundary	Knowledge has context
Schema	Self-describing type definition	Structure is extensible
Identity	Ed25519 keypair, self-referential	Attribution is cryptographic

One primitive. Everything else is a content type.

Ergo cognito sum.