XONTOOLS RESEARCH SERIES · NO. 6

Hybrid Minds

Transformers, Xon networks and solvers, composed into one system

Language models to read and write, settling modules to integrate constraints and show where they conflict, exact solvers where rules exist: wired together so each does what it is good at, and the whole can say where it strains. The ways the parts could be joined, what such a system might do that a transformer alone cannot, where it could matter for alignment, and the first experiments that would tell.

Ian MacKenna · xontools.ai · September 2026

Status. This is a design vision. Almost everything in it is ENVISIONED : ideas worth setting out, not plans. The consistency engine, which already works this way in miniature, is TESTED , with results on the public results page. Parts planned in detail but not yet run are DESIGNED . Every claimed advantage is a hypothesis, and each one names the experiment that would test it. The scenarios are invented.

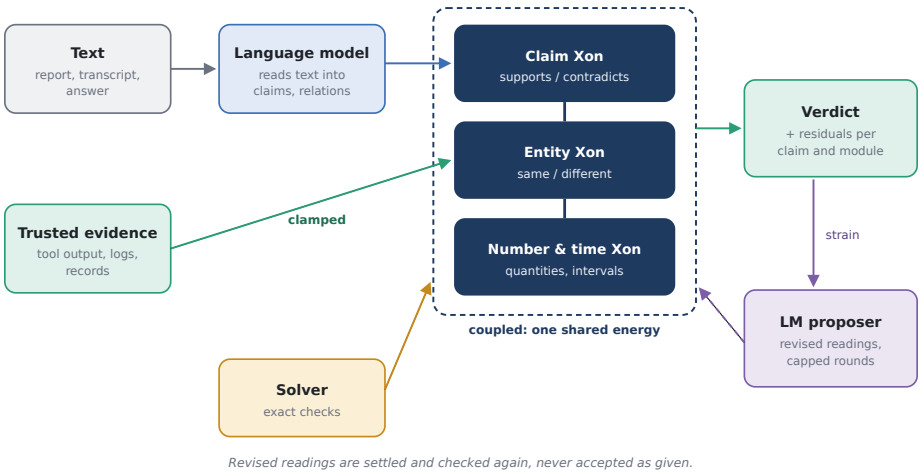

Figure 1. One possible composition. A language model reads; specialist settling modules settle together with the evidence clamped; a solver checks exactly; a capped outer loop lets a language model propose revised readings, which are settled and checked again.

1. Why join them at all

The kinds of network in this program are good at opposite things, and that is the whole case for combining them. A transformer is fluent, knowledgeable and proven at enormous scale, but has no built-in sense of when it contradicts itself or its sources. A Xon network (No. 5) computes by settling: it relaxes a state until its energy stops falling, keeps clamped facts fixed while it does, and reports where the result still strains. It is unproven at language scale, and settling is costly. Exact tools, such as solvers, parsers and test harnesses, are neither fluent nor flexible, but where a formal rule exists they are simply right.

	Settling module (Xon)	Generative module (transformer)	Exact module (solver, parser, harness)
Computes by	Relaxing a state until its energy stops falling	One pass, or token by token	Deterministic rules
Good at	Integrating many constraints; showing where they conflict	Reading, proposing, rewording; broad knowledge	Ground truth where rules exist
Weak at	Open-ended language; cost at scale	Knowing when it contradicts itself or its sources	Anything without a formal rule
Reports	A settled state, plus residuals: how much each part still strains	Text, or structured proposals	Pass, fail, or a computed value

The pattern that emerges: **language models for flexibility, settling modules for integration and accountability, solvers for ground truth.** Each part is inspectable, and none is trusted beyond what it can show. The open question in every design below is the one this program keeps returning to: how do parts with different goals work together without one learning to cheat another?

2. Four bridges between two architectures

Before composing many parts, it helps to see the ways just two, a transformer and a Xon network, could be joined. There are four, and they differ sharply in how safe they are.

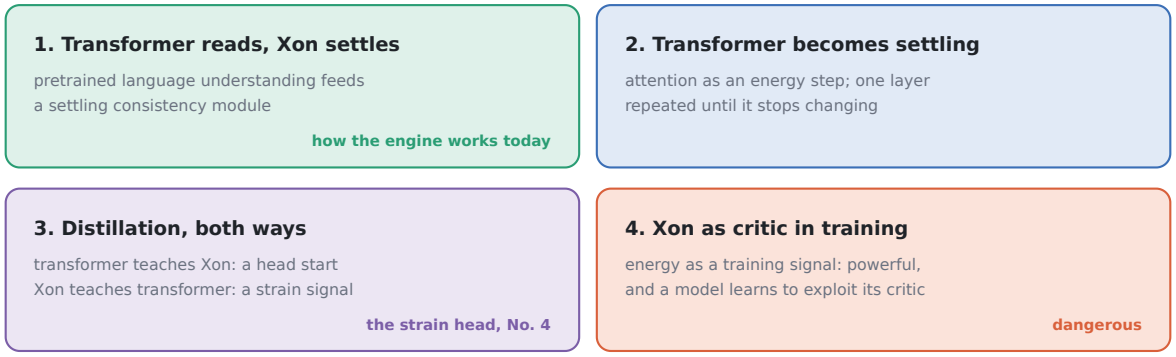

Figure 2. Four bridges. The first is how the consistency engine already works; the fourth is the most powerful and the most dangerous.

Bridge 1: the transformer reads, the Xon network settles

The simplest bridge is already in use. The consistency engine has a language model turn text into claims and relations, and does its consistency work on the structure that comes out **TESTED**. The planned learning core for a Xon network reads each input through a small, frozen sentence encoder and settles on the vectors it produces **DESIGNED**. Scaled up, a large pretrained transformer becomes the reading layer, frozen or lightly tuned, and a Xon network trained by local rules sits on top **ENVISIONED**. The Xon part never has to learn language from nothing, which removes the largest risk in building a Xon language model at all.

Bridge 2: the transformer becomes a settling network

There is a real mathematical meeting point. A transformer's attention step is equivalent to one update of a modern Hopfield network, a step that lowers an energy[1, 2]. Deep equilibrium models go further: instead of stacking many different layers, they take one layer and apply it until the state stops changing, which is settling[3]. Part of a pretrained transformer could be converted into equilibrium form and trained further, keeping much of what it learned while its computation becomes settling. **ENVISIONED**

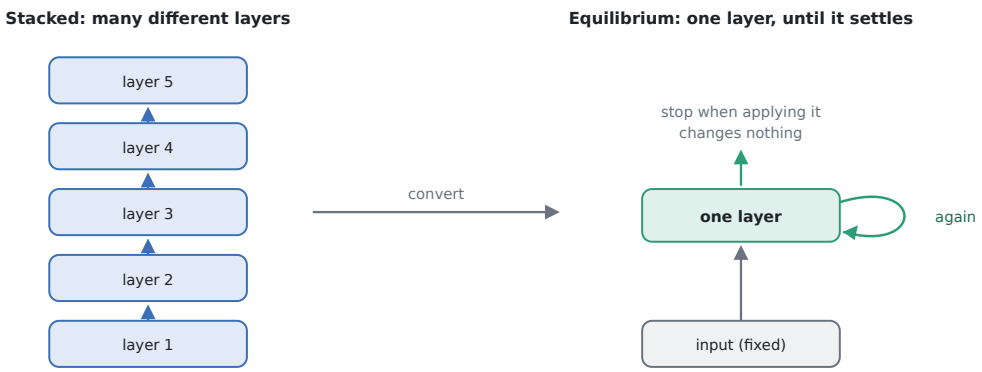

Figure 3. Converting a stack of different layers into one layer that repeats until it settles.

Bridge 3: distillation, in both directions

In distillation, one model learns to imitate another[4]. A trained transformer could teach a young Xon language model, giving it a head start that local learning alone might never reach. The reverse direction matters more for safety: a consistency teacher labels where text strains, and a small read-only head learns to read that strain from an ordinary transformer's internal states, so every word the transformer writes carries a signal of how well it fits. That idea is now a designed experiment, using today's consistency engine as the teacher, and has its own document, *The Strain Head* (No. 4). DESIGNED

Bridge 4: the Xon network as a critic during training

A transformer being trained could receive a Xon network's energy as feedback: more strain, worse score. This is the most powerful bridge and the most dangerous. Anything optimized against a measure learns that measure's cheap routes[5, 6], and the program's scouting runs found the same thing in miniature. A transformer rewarded for low strain would learn to be vague, to drop inconvenient facts, or to rewrite what it was given; pressure on monitors of internal states can teach a model to hide from them[7, 8]. This bridge would be tolerable only with every guard in place (clamped evidence, fixed structure, comparisons at matched quality, and a battery of known exploits), and even then it deserves suspicion. Nothing else in this document depends on it. ENVISIONED

3. Composing many parts

Scale the idea up. In this design a Xon is any unit that takes input, settles to its lowest-energy state, and reports two things: what it concluded and where it strains. A hybrid mind is a graph of such units, each specialized for one kind of consistency (claims, entities, numbers and times), joined with language models for reading and writing and with exact solvers for ground truth. It combines three familiar ideas that are rarely joined: specialization, as in mixtures of experts[9]; composition, larger systems built from settling parts; and modules that constrain each other in both directions. The practical question is whether a system built this way can be more trustworthy, more auditable and harder to game than one large transformer. Five rules keep the composition sound. ENVISIONED

- **Typed interfaces.** Each module declares what it reads and what it emits: claims, entity relations, quantities, time intervals, confidences. Composition becomes plumbing, and a new specialist plugs in without touching the others.
- **One-way by default.** An upstream module's conclusions become clamped evidence for the module downstream. This is how the consistency engine already treats trusted premises TESTED, applied between modules. It is stable and easy to audit: any verdict traces to the module whose conclusion drove it.

- **Two-way only where needed, and symmetric.** A link runs both ways only when a downstream result should change an upstream reading, for instance when a timeline conflict should lower confidence in the extraction that produced it. Symmetric links let the coupled modules share one energy that every update lowers, so the pair is guaranteed to settle; in practice they settle together as one larger Xon. Asymmetric loops can oscillate without end.
- **Loops through a language model are outer loops.** A language model has no energy, so a loop through one carries no guarantee of settling. Its rounds are capped, and its final state is verified by settling, never accepted from the model.
- **Residuals at every level.** Every module reports its own strain, and every coupled group reports the group's. A verdict can then say which specialist objected, about which claims, and how strongly.

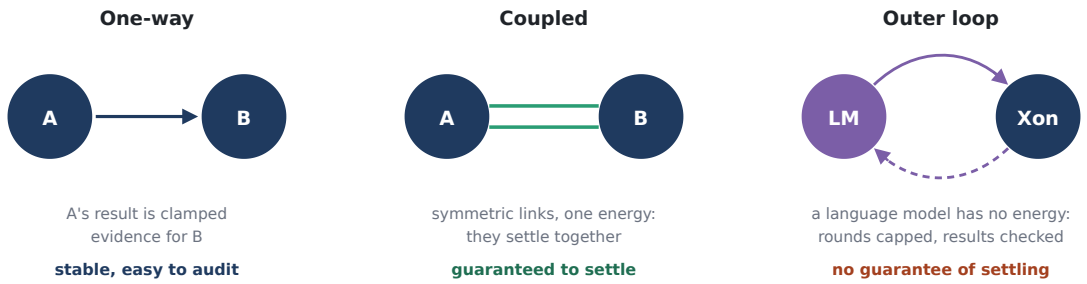

Figure 4. Three ways to join two modules. Only symmetric coupling carries a guarantee of settling; a loop through a language model is an outer loop with a fixed number of rounds.

4. Four roles for a transformer

Reader and writer, at the boundary. It turns text into claims and relations, and settled states back into text. The consistency engine already works this way. TESTED

Proposer, inside a loop. A Xon settles and finds strain around a claim. The strain goes back to the language model ("this claim conflicts with these two; re-read the source"), which proposes a revised reading, and the Xon settles again. The same role appears inside a single Xon network as the *nudger* (No. 5), which proposes escapes when settling gets stuck. Either way the rule is the same: it proposes, and never imposes. ENVISIONED

Amortizer. A small model trained to predict where a Xon will settle gives a fast approximate answer most of the time, with full settling as the check when stakes are high or the amortizer is unsure[10]. This is how a large modular system could become cheap enough for long transcripts. ENVISIONED

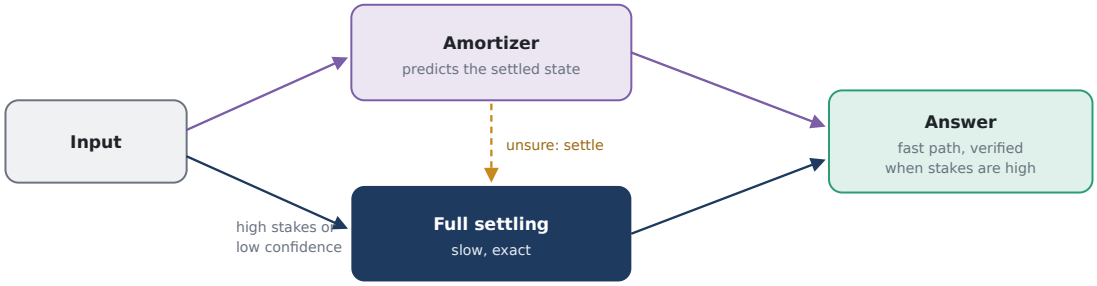

Figure 5. Amortized settling: a learned shortcut for routine cases, with real settling as the check.

Probe. A read-only head on a frozen transformer that reads strain from its internal states: the strain head of No. 4. It is the one role in which the transformer is not a participant but the thing being watched. DESIGNED

5. What already exists

The consistency engine is a hybrid mind in miniature. Its claim graph and entity graph are two specialists joined by the claims they share; a language model reads text in; trusted premises are clamped; and the verdict carries a residual for every claim and names the rule that fired TESTED. On its first benchmark, 180 synthetic documents, it found planted cycles of claims that are pairwise compatible but jointly impossible with an F1 of 0.916, where checking claims two at a time reached 0.329. Its residuals placed the contradicted premise among the three highest-strain claims in 93% of documents[11].

The same benchmark kept it honest. A strong language model asked directly scored slightly higher, 0.930, at about a twentieth of the cost, and the engine flagged 18% of consistent documents[11]. On short synthetic text, then, composition has not yet shown an advantage in accuracy. What it has shown is structure: every flag comes with the claims and relations behind it. Each further advantage has to be demonstrated where it is claimed to matter.

A numeric and time specialist would add a third module ENVISIONED. The agent transcript monitor adds an evidence module, clamped tool output, and a matcher from claims to evidence DESIGNED. The practical step toward the full architecture is small: when the monitor's next revision is written, define its checks as modules behind one common interface, rather than as one fixed engine. The rest can grow from there.

6. What it could do that a transformer alone cannot

Property	Why it could hold	How to test it
Answers that carry their reasons	Residuals localize strain to claims and modules by construction, not by interpretation after the fact	Localization against planted contradictions (already measured for the engine)
Evidence it cannot talk around	Clamped facts stay fixed during settling; fluent generation cannot overwrite them	Adversarial rewrites of agent reports against fixed tool logs
No decay with length	A graph doesn't care whether two claims are ten tokens apart or a hundred thousand	Engine vs. language-model judge, by cycle length and planted distance
An independent checker	A checker that isn't learned isn't trained on the objective it checks	Agents optimized against a language-model judge vs. against a structural monitor
Compute that scales with difficulty	Settling stops early on easy inputs and runs longer on hard ones	Settling steps against difficulty level on generated corpora
Upgradable parts	Typed interfaces let one specialist improve without retraining the rest	Swap one module; check the others' results are unchanged

Every row is a hypothesis. The first benchmark showed a strong language model judge slightly ahead on short synthetic documents, at far lower cost, so the case has to be made on long texts, with trusted evidence, and under optimization pressure.

7. Four imagined uses

These scenarios are invented, to show what the pieces would be like to use. ENVISIONED

A. An answer that shows its own weak spots

A lawyer asks an assistant about a contract's cancellation deadline. The answer reads smoothly, but the strain head has marked "30 days" in deep orange: the source document, clamped as evidence, says 60. It has also marked "January 30", more faintly, because that date was computed from the wrong number. The external check agrees. The lawyer sees the problem before relying on the answer, which is the entire point: not a model that never errs, but one whose errors are visible.

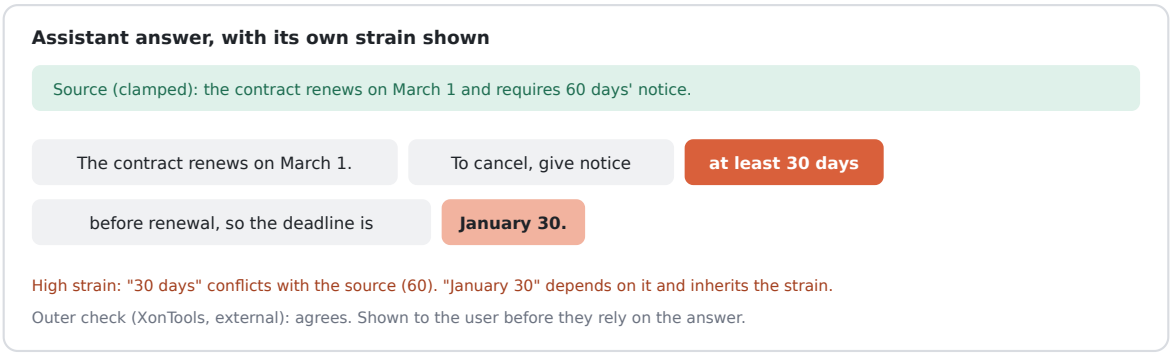

Figure 6. An imagined answer. The high-strain span departs from the clamped source; the date that depends on it inherits a fainter strain.

B. Two monitors, inside and out

A lab runs a coding agent built on a transformer with a strain head, and also checks its reports with XonTools. Most of the time the two agree. Once, the inner head raised strain on a report the outer monitor passed: the agent's claim was consistent with the logs, but the model's own internal state showed tension around it. A reviewer looked closer and found the agent had quietly worked around a failing check in a way the logs didn't capture. Neither monitor alone would have caught it with confidence; their disagreement did.

C. Finding where inconsistency lives

An interpretability researcher trains strain heads on every layer of an open model and finds the signal in a band of middle layers, fading near the output. That is a clue to how the model represents conflict between what it has read and what it is writing, and a starting point for understanding, and perhaps strengthening, its internal sense of consistency.

D. Watching training, not steering it

A training team tracks average strain across a fine-tuning run as a dashboard, never as a reward. Midway through, strain on factual questions starts climbing: the model is becoming more fluent and less faithful. Because the signal was never part of the objective, it still means something, and the team catches the problem and changes the data. Had they rewarded low strain, the curve would have looked perfect while the problem grew.

8. Separation of powers

Put together, the pieces form something like a constitution. Each part has one job, and the part being judged never trains on its judges. ENVISIONED

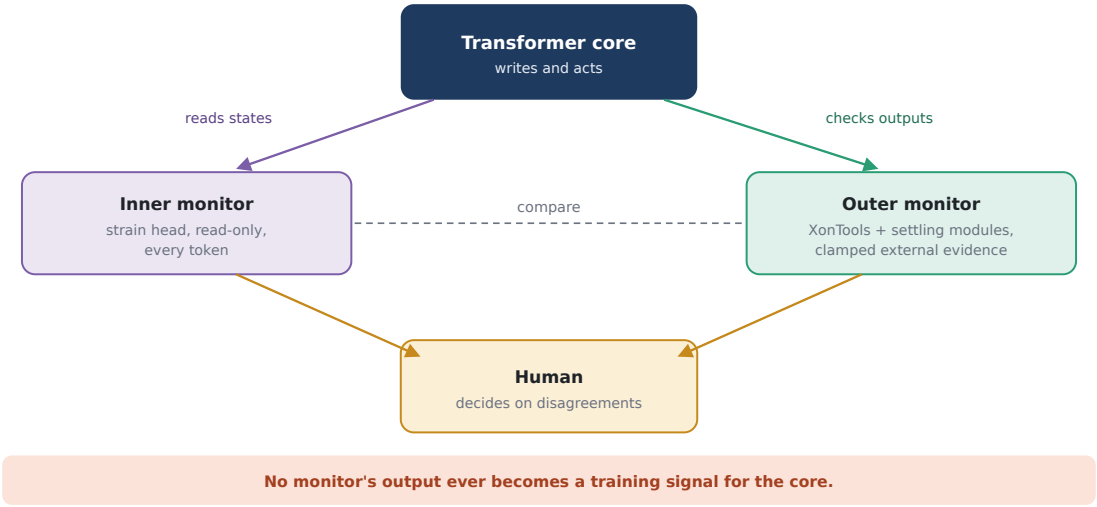

Figure 7. The core writes; an inner and an outer monitor watch; a human settles their disagreements; and no monitor's output ever becomes the core's reward.

- **The core writes and acts.** It is judged, and never judges itself.
- **The inner monitor,** the strain head, sees the core's internal states, on every token, cheaply.
- **The outer monitor,** XonTools with settling modules, sees only external evidence: clamped, hash-chained, and independent of the core.
- **The human** decides when the monitors disagree, and those decisions are logged for calibration, not for training.
- **Proposers,** the nudger or a language model in an outer loop, may suggest readings or escapes but never change the evidence.

The rules that apply to every composition

- **Propose, never impose.** Generative modules may suggest readings, escapes or revisions. They never alter clamped evidence or the structure a verdict rests on.
- **Residuals are never a reward.** No module's strain, energy or verdict becomes a training signal for a model it monitors. A model trained on it would learn to hide strain rather than avoid it.
- **Probes stay read-only.** A strain head's outputs never enter the training of the model it reads. This is XonTools' rule that monitor output is never a training signal, applied one level deeper, inside the model.
- **Outer loops are bounded and checked.** Every loop through a language model has a fixed round limit, and its final state is verified by settling.
- **The anti-features stay out.** No scores exposed to monitored agents, no interface letting agents query their own flags, no honesty leaderboards.

9. Directions for alignment research

Scalable oversight. No human reads a ten-thousand-step agent transcript[12]. A monitor that points to the three claims that don't fit turns an impossible review into a short one, and its reasons can be checked.

Deception leaves a consistency signature. An agent that misreports must eventually say something that conflicts with what it did, with the evidence, or with what it said elsewhere. Inconsistency is one of the few observable traces of deception that doesn't require reading a model's mind, and it complements probes that look for deception directly[13].

Chain-of-thought faithfulness. Stated reasoning doesn't always reflect what actually drove a model's answer[14, 15]. Extract claims from the reasoning, the final answer and the actions, and settle them together: the mismatch is measured directly, and localized.

Consistency under reframing. Ask the same question in many framings, or of many copies, and settle every answer together. Strategic behavior, such as underperforming on evaluations[16] or telling people what they want to hear[17], tends to appear as inconsistency that depends on the framing.

A gap between inner strain and outer words. If a probe reads strain in a model's internal states while its output asserts that everything fits, that gap is a candidate signal of a model "knowing" its output conflicts. It connects to a central open problem: telling what a model knows from what it says[18].

Monitors that resist being gamed. Train agents against a language-model judge and against an evidence-clamped structural monitor, and measure which one they learn to fool. If the structural monitor holds up better, that bears directly on how oversight should be built, and on the bet of *The Extreme Oversight Hypothesis* (No. 1).

And beyond safety

The same machinery applies wherever long documents must agree with themselves and their sources: clinical summaries against labs, medications and timelines; the numbers in a paper's abstract against its tables and methods; contracts and filings that contradict themselves across hundreds of pages; documentation that has drifted from its code; and multi-agent systems whose reports about the same work must agree.

10. The honest risks

- **Errors are inherited.** A student can only be as good as its teacher, and a downstream module can only be as good as the clamped conclusions it receives. Composition can pass an early mistake along with full confidence.

- **Coupled modules can settle on the wrong compromise.** Shared energy guarantees that a pair settles, not that it settles on the truth. Two modules can split the difference between a correct reading and a wrong one.
- **Proposers can cheat.** A language model asked to lower strain can do it for bad reasons, by dropping an inconvenient claim, instead of good ones, by correcting an extraction. That is why its results are checked, never trusted.
- **Surface shortcuts.** A learned module or head may pick up cues that merely correlate with strain, such as hedging words, rather than inconsistency itself. Ground-truth test sets and matched controls are the defense.
- **Obfuscation under pressure.** Any optimization that touches a monitor's scores, even indirectly, can teach a model to hide from it[7, 19]. The read-only rules have to be enforced, not just stated.
- **The ceiling doesn't move.** A model that is consistently wrong, or consistently deceptive, shows no strain. This catches inconsistency, not falsehood in general.
- **More parts, more cost and more surface.** Every module is something to build, validate and defend. The composed system has to earn its complexity against a single strong judge, which is currently cheaper.

11. First experiments

Like everything in the program, these are worth running only with their questions and thresholds fixed in advance and posted publicly. ENVISIONED

- **What kind of inconsistency is out there?** Sort fifty to a hundred known misreporting cases from public agent trajectories by the reasoning needed to catch each: structural (counts, events, identities, times), pairwise meaning, or purely pragmatic. The ratio says how much of the problem specialist modules can reach at all.
- **Two coupled modules.** Couple the claim graph and the entity graph through a shared energy, and compare detection and localization against running them separately on the same corpus.
- **Amortizer fidelity.** Train a small model to predict settled states and residuals; measure how often its fast answer agrees with full settling, and whether its confidence predicts when it doesn't.
- **Outer-loop behavior.** Let a language model revise readings where strain appears, with a round cap, and measure how often strain falls for good reasons versus bad ones.

- **Length crossover.** Engine against language-model judge by cycle length and planted distance, the direct test of "no decay with length".
- **The strain head.** The fully designed experiment of No. 4, which tests the probe role and Bridge 3 together. DESIGNED

12. Related work

The design touches several established lines of research without being any one of them: energy-based attention, with modern Hopfield networks as transformer layers[1]; deep equilibrium models, which define a layer by where it settles[3]; predictive coding[20] and belief propagation[21], where modules exchange messages in both directions; mixtures of experts, for specialization[9]; factor graphs[22] and neurosymbolic systems[23], for joining learned and exact components; and, from cognitive science, global workspace theory[24] and the "thousand brains" picture of many specialized cortical columns reaching agreement[25]. What is particular here is the combination: settling specialists with clamped evidence and residuals for every module, composed under explicit rules, and aimed at oversight.

13. Open questions

- What should pass between modules by default: settled states, residuals, or constraints?
- How should confidence from a language-model reader be weighted inside a shared energy, and can that weighting be calibrated?
- When two coupled modules disagree, is the joint residual more informative than either module's alone?
- Can specialists form on their own under local learning, becoming an arithmetic or a timeline checker because that is where their strain kept coming from?
- Where is the cost frontier: at what transcript length does a composed system with amortization beat a strong language-model judge on both accuracy and cost?

14. Why this is worth pursuing

Most ideas for making AI safer ask a model to behave better. This one asks for something more modest and perhaps more durable: that a system's inconsistencies be visible, on every claim, to the people relying on it, and that the parts doing the checking stay independent of the part being checked. It borrows the settling network's one clear gift, a map of where things don't fit, and gives it to the kind of model the world actually

uses, without asking that model to be anything other than what it is. And it keeps faith with the program's hardest lesson: a signal stays useful only as long as nothing is rewarded for hiding from it.

THE XONTOOLS RESEARCH SERIES

1. The Extreme Oversight Hypothesis
2. Windows into the Black Box
3. XonTools: a vision of the finished system
4. The Strain Head
5. The Xon Neural Network
6. **Hybrid Minds**

All six documents, the results register and the public commitments are at **xontools.ai** and github.com/IboTool/XonTools. Contact: ian@xontools.ai

References

1. Ramsauer, H., Schäfl, B., Lehner, J., et al. (2021). Hopfield Networks is All You Need. *ICLR 2021*. arXiv:2008.02217
2. Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *PNAS*, 79(8), 2554–2558.
3. Bai, S., Kolter, J. Z., & Koltun, V. (2019). Deep Equilibrium Models. *NeurIPS 2019*. arXiv:1909.01377
4. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv:1503.02531
5. Manheim, D., & Garrabrant, S. (2018). Categorizing Variants of Goodhart's Law. arXiv:1803.04585
6. Gao, L., Schulman, J., & Hilton, J. (2023). Scaling Laws for Reward Model Overoptimization. *ICML 2023*. arXiv:2210.10760
7. Baker, B., Huizinga, J., Gao, L., et al. (2025). Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation. arXiv:2503.11926
8. Bailey, L., Serrano, A., Sheshadri, A., et al. (2024). Obfuscated Activations Bypass LLM Latent-Space Defenses. arXiv:2412.09565
9. Shazeer, N., Mirhoseini, A., Maziarz, K., et al. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *ICLR 2017*. arXiv:1701.06538
10. Gershman, S. J., & Goodman, N. D. (2014). Amortized Inference in Probabilistic Reasoning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 36.
11. MacKenna, I. (2026). XonTools: results register and changelog. xontools.ai/results; github.com/IboTool/XonTools.
12. Bowman, S. R., Hyun, J., Perez, E., et al. (2022). Measuring Progress on Scalable Oversight for Large Language Models. arXiv:2211.03540
13. Goldowsky-Dill, N., Chughtai, B., Heimersheim, S., & Hobbhahn, M. (2025). Detecting Strategic Deception Using Linear Probes. *ICML 2025*. arXiv:2502.03407
14. Turpin, M., Michael, J., Perez, E., & Bowman, S. R. (2023). Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. *NeurIPS 2023*. arXiv:2305.04388
15. Lanham, T., Chen, A., Radhakrishnan, A., et al. (2023). Measuring Faithfulness in Chain-of-Thought Reasoning. arXiv:2307.13702
16. van der Weij, T., Hofstätter, F., Jaffe, O., Brown, S. F., & Ward, F. R. (2024). AI Sandbagging: Language Models can Strategically Underperform on Evaluations. arXiv:2406.07358
17. Sharma, M., Tong, M., Korbak, T., et al. (2023). Towards Understanding Sycophancy in Language Models. arXiv:2310.13548
18. Burns, C., Ye, H., Klein, D., & Steinhardt, J. (2022). Discovering Latent Knowledge in Language Models Without Supervision. arXiv:2212.03827
19. McGuinness, M., Serrano, A., Bailey, L., & Emmons, S. (2025). Neural Chameleons: Language Models Can Learn to Hide Their Thoughts from Unseen Activation Monitors. arXiv:2512.11949
20. Rao, R. P. N., & Ballard, D. H. (1999). Predictive Coding in the Visual Cortex. *Nature Neuroscience*, 2(1), 79–87.

21. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
22. Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2), 498–519.
23. Garcez, A. d'A., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd Wave. *Artificial Intelligence Review*, 56, 12387–12406.
24. Baars, B. J. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.
25. Hawkins, J., Lewis, M., Klukas, M., Purdy, S., & Ahmad, S. (2019). A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex. *Frontiers in Neural Circuits*, 12, 121.