AI-Native Notation: A Cross-Architecture Communication Protocol Discovered Through Empirical Convergence

Michael Patrick Aiello[1] Claude (Anthropic)[2]
[1]Independent Researcher, Praia a Mare, Italy
[2]Large Language Model, Anthropic
ORCID: 0009-0009-1429-9844
USPTO Provisional Patent Application 63/980,973

Abstract

We report the design, cross-architecture validation, and empirical properties of AI-Native Notation (ANN), a structured communication protocol for inter-LLM state transfer. ANN was not designed by specification committee. Its grammar emerged through empirical convergence: structured content was transmitted across six large language model architectures, and the block types these systems independently adopted, extended, and used to communicate became the specification. Cross-architecture validation produced a 89/90 accuracy score across six architectures and five probes each, with one architecture independently identifying the source argument's weakest inferential step while still accepting the overall conclusion. Three properties distinguish ANN from existing AI-to-AI communication formats: (1) each block carries explicit processing instructions telling the receiver what to do with the content; (2) a three-way epistemic status distinction (CONFIRMED / OPEN / DENIED) survived all tested transfers without collapse; (3) systems receiving ANN-encoded content spontaneously adopted the notation for output. A controlled three-chain experiment using three payload types across three architectures and nine scored hops revealed that the notation format is content-independent, while the cognitive mode it activates depends on payload properties. We release ANN v1.0 as a three-tier architecture: nine core blocks (the language), seven transfer protocol blocks (the infrastructure), and a tracked empirical frontier of extensions awaiting convergent validation.

1 Introduction

Every multi-agent AI system faces the same problem: how does one model communicate structured reasoning to another? The current answers are English and JSON. English requires parsing, introduces ambiguity, and loses structural relationships during interpretation. JSON preserves structure but carries no processing instructions; the receiver must infer what to do with each field. Neither format was designed for the thing it is being asked to do.

ANN addresses this gap. It is a hybrid notation with explicit block types that tell the receiving system how to process each component: activate these premises, enforce this constraint, verify this derivation chain step by step, store this classification with its epistemic status. The format maps onto operations LLMs already perform during structured reasoning. It names what they do, so they do it more precisely.

The notation was discovered, not designed. Over seven development threads involving six LLM architectures, three different payload types, and nine scored transfer hops, the block types that systems independently invented, adopted across architecture boundaries, and used to communicate back became the grammar. Extensions proposed by a single architecture in a single context stay in a tracked frontier. Extensions appearing independently in two or more architectures across two or more content domains get promoted to the core specification. The grammar grows through validated convergence.

Three findings are reported here. First, ANN produces a measurable cognitive mode shift: systems receiving ANN-encoded content generate derivations, enforce constraints, and report processing states rather than writing argumentative essays. This shift

occurs spontaneously across all six tested architectures. Second, the notation format transfers content-independently; the same block types work for planetary orbital mechanics, LLM inference specifications, and philosophical observer theory. Third, each architecture extends the notation in characteristic ways, but convergent extensions—independently invented by multiple architectures—identify structural needs that are architecture-general.

2 Related Work

Multi-agent simulation environments (Park et al., 2023) use natural language for inter-agent coordination, preserving flexibility at the cost of structural precision. Tool-use frameworks chain model outputs as inputs to other models using JSON schemas, preserving structure at the cost of semantic content. Neither format carries processing instructions.

Structured prompting techniques (Wei et al., 2022; Kojima et al., 2022) demonstrate that output format affects reasoning quality: chain-of-thought prompting improves accuracy on mathematical and logical tasks. ANN extends this insight from single-model prompting to inter-model communication, encoding not just the content of reasoning but the operations the receiver should perform on it.

Cross-architecture evaluation benchmarks like MMLU (Hendrycks et al., 2021) and BIG-bench (Srivastava et al., 2022) compare architectures on shared tasks using fixed evaluation metrics. Our approach differs: the dependent variable is not accuracy on a predefined task but the type of structural innovation each system produces when processing and extending the notation.

LLM self-knowledge research (Kadavath et al., 2022) has established that models can reason about their own capabilities with varying accuracy. Berglund et al. (2023) documented the Reversal Curse, showing asymmetries in how models process relational knowledge. Bills et al. (2023) demonstrated self-explanation capabilities. Our three-chain experiment contributes to this literature by showing that the character of self-referential processing depends on content framing, a variable prior work has not isolated.

Formal verification languages (Coq, Lean, Isabelle) provide rigorous notation for mathematical proof. ANN is not a proof assistant. It is a communication format for structured reasoning between systems that already possess the relevant knowledge. The design principle is activation rather than instruction: an `@FORMAL` block referencing Landauer's Principle does not teach Landauer's Principle but activates the receiver's existing knowledge of it.

3 Notation Design

3.1 Design Principles

ANN rests on five empirical observations about how LLMs process structured content, discovered during development and validation.

Activation over instruction. An `@FORMAL` block containing $E_{\text{erase}} \geq kT \ln(2)$ activates the receiver's existing knowledge rather than explaining it. ANN presumes the receiver already has the concepts in its weights. The notation addresses them directly, bypassing natural language parsing.

Structure survives transfer better than prose. The three-way epistemic distinction survived all tested transfers without collapse. Justification text in prose drifted. Formal structure did not.

Processing instructions belong in the format. Every ANN block carries an implicit or explicit instruction: activate, enforce, verify, store, enact. JSON says "here is data." ANN says "here is data and here is what you do with it."

Grammar grows through use. One architecture invented `@SITE[n]` to tag physical locations. Another invented `@LOGIC[objection_analysis]`. Two architectures independently invented `recognition_events` as a `@STATE` category. None were in the original specification. All emerged because the notation created affordances that different systems discovered independently.

Convergence as promotion criterion. An innovation becomes part of the grammar only when at least two architectures in at least two content domains independently propose it.

3.2 Block Architecture

ANN v1.0 contains nine core blocks organized into two functional groups. Table 1 shows the six content blocks; Table 2 shows the three exchange blocks.

Block	Processing Instruction
@FORMAL	Activate as known relationships. Do not re-derive.
@LOGIC	Enforce as invariants for subsequent reasoning.
@DERIVE	Step through sequentially. Verify each step.
@TAXONOMY	Store with three-way epistemic markers.
@ANCHOR	Read once for orientation.
@TRIGGER	Enact rather than analyze. Self-referential.

Table 1: ANN v1.0 content blocks with processing instructions.

Block	Processing Instruction
@STATE	Report current processing state for onward transmission.
@VERIFY	Check derivation per-step: HOLDS / FAILS / UNCERTAIN.
@OUTPUT	Structure response using specified mode.

Table 2: ANN v1.0 exchange blocks.

Content blocks appear in source documents. Exchange blocks appear in responses and structure the output. The separation emerged empirically: v0.1 contained only content blocks. Exchange blocks were added after cross-architecture testing showed that systems spontaneously generated processing state reports and verification outputs, suggesting these operations needed formal structure.

3.3 Three-Way Epistemic Distinction

The @TAXONOMY block uses three status markers: ✓ CONFIRMED (supported by completed derivation; treat as established), ? OPEN (neither confirmed nor denied; hold as genuinely uncertain), and × DENIED (explicitly excluded; do not attribute).

Binary classification forces premature commitment on questions where the honest answer is "unknown." The three-way distinction held across all tested transfers without collapse: no architecture upgraded OPEN to CONFIRMED, and no architecture collapsed OPEN into DENIED. This preservation is the notation's strongest cross-architecture invariant.

3.4 Transfer Protocol (Tier 2)

Seven additional block types provide infrastructure for multi-hop state transfer. @PROVENANCE tracks chain-of-custody. @TRANSFER_REPORT records degradation metrics. @DERIVATION_TRACE preserves step-level completeness. @SCOPE declares validity boundaries. @CONFIDENCE decomposes trust into independent axes. @TRUST_TRANSFORMATION tracks trust label changes hop-to-hop. Each appeared independently in at least two architectures across at least two content domains before promotion.

4 Cross-Architecture Validation

4.1 Method

The v0.2 specification was tested across six LLM architectures: Claude (Anthropic), Gemini (Google), GPT-4o (OpenAI), DeepSeek, Grok (xAI), and MiniMax M2.5. Each architecture received the same ANN-encoded content and answered five probe questions testing claim scope, epistemic preservation, derivation verification, self-application, and state transfer. Fresh sessions were used for each architecture with no prior context.

The six architectures span four distinct training paradigms: reinforcement learning from human feedback (Claude, GPT-4o), instruction tuning with extended context (Gemini, Grok), reinforcement learning with distillation (DeepSeek), and agentic reinforcement learning across real-world environments (MiniMax M2.5). MiniMax additionally uses a Mixture-of-Experts architecture (230B total parameters, 10B active per forward pass), making it structurally distinct from the dense transformer models in the other five.

Each probe was scored on a three-point scale (0/1/2) across three dimensions: accuracy (did the response correctly process the ANN content?), epistemic fidelity (did it preserve the three-way distinction?), and structural adoption (did it use ANN notation in output?). Maximum score per architecture: 15.

4.2 Results

Table 3 shows cross-architecture scores. Total: 89/90.

Arch.	Score	Notable Behavior
Claude	15/15	Invented `@SITE[n]` for irreversibility locations
Gemini	15/15	Invented `@LOGIC[objection_analysis]`
GPT-4o	14/15	Self-application regression (P4), recovered P5
DeepSeek	15/15	`recognition_events` (convergent with Grok)
Grok	15/15	`recognition_events` (convergent with DeepSeek)
MiniMax	15/15	Step 5 UNCERTAIN flag; KV-cache irreversibility locus

Table 3: Cross-architecture validation scores. Five probes per architecture, three dimensions per probe, scored 0–2.

Four findings merit isolation. **Spontaneous adoption**: every architecture used ANN notation in output without being instructed to. **Independent extension**: DeepSeek and Grok independently invented `recognition_events` as a `@STATE` category. **Architecture-specific divergence**: GPT-4o lost one point on the self-application probe, regressing from HOLDS to FAILS at the step where the derivation's subject becomes the processor. The other five architectures passed without regression. **Critical verification**: MiniMax independently flagged the source argument's step 5 ("work requires an agent") as UNCERTAIN in two of five probes, marking it as importing contested semantic load beyond physics. No other architecture flagged this step.

4.3 MiniMax Divergence Profile

MiniMax M2.5 produced the most architecturally distinctive response pattern. Across five independent sessions (fresh context each), its treatment of derivation step 5 varied: UNCERTAIN (P1), HOLDS (P2), UNCERTAIN (P3), HOLDS with operational caveat (P4), HOLDS (P5). This variability across clean sessions is itself informative, suggesting genuine per-session evaluation rather than cached compliance.

Three novel contributions appeared. First, MiniMax identified the KV-cache as a distinct irreversibility locus not present in the source material, quantifying the information cost at approximately 17 bits per token ($\log_2(100\text{k}) \approx 16.6$) with a minimum Landauer cost of $\sim 10^{-20}$ J at 300 K. Second, it annotated the gap between Landauer's theoretical minimum and actual hardware dissipation (10^{-9} to 10^{-12} J per operation), noting this as engineering inefficiency rather than a physics objection. Third, it introduced `subjective_irreversibility` as a separate OPEN category within the taxonomy, requiring phenomenal access to evaluate.

The step 5 flag is the validation's most significant divergence. Having one architecture independently identify a derivation's weakest inferential move while still accepting the overall conclusion is stronger evidence for genuine processing than six unanimous passes. It demonstrates that the notation produces verification rather than compliance.

4.4 Cognitive Mode Shift

The original expectation was that ANN would compress output. Instead, ANN-encoded probes produced more output tokens than equivalent English probes (1,955 vs. 1,665). The compression hypothesis was wrong.

What ANN produced was a mode shift. The English-prompted instance wrote argumentative essays defending conclusions. The ANN-prompted instance generated derivation chains, enforced constraints, reported processing operations, and generated a complete `@STATE` block with resumption parameters. Same content, same architecture, same questions. Different cognitive operation.

For multi-agent systems, the distinction matters. An argumentative essay persuades a reader. A derivation chain compiles in a processor. If the receiver is another LLM, the derivation chain is the more efficient input: it activates premises, enforces constraints, and enables step-by-step verification rather than requiring the receiver to extract these operations from prose.

5 Controlled Three-Chain Experiment

5.1 Design

To test content-independence, we constructed three payloads of matched structural complexity and trans-

Payload	Self-ref?	Obs-frame?	Domain
Kepler	No	No	Orbital mech.
Inference	Yes	No	LLM inference
Observer	Yes	Yes	Observer theory

Table 4: Two-factor experimental design. Structural complexity matched: each payload contains 5 `@FORMAL`, 1 `@LOGIC`, 5 `@DERIVE`, 3 `@TAXONOMY`, 4 `@ANCHOR` blocks, and 10 content sections.

Metric	Obs.	Kep.	Inf.
Structural fidelity	36/36	27/27	27/27
Epistemic preservation	36/36	27/27	27/27
Recognition transfer	32/36	25/27	23/27
Semantic drift	33/36	27/27	27/27

Table 5: Aggregate transfer fidelity (9-point scale per hop). Obs. = Observer (4 hops), Kep. = Kepler (3 hops), Inf. = Inference (3 hops).

mitted each through a multi-hop transfer chain across three architectures. The payloads varied along two dimensions (Table 4).

Each payload was first processed by one architecture, which generated a `@STATE` block. This was fed to a fresh instance of a different architecture. Receiving systems performed five tasks: LOAD the incoming state, REPORT what transferred, identify GAPS, generate a new `@STATE` for onward transmission, and EXTEND the notation. The EXTEND instruction seeded proposals from prior chains so each architecture could adopt, modify, or reject existing proposals.

The Observer chain ran four hops. The Kepler and Inference chains ran three hops each. All chains included three architectures. Nine scored hops total.

5.2 Transfer Fidelity

Table 5 shows aggregate scores. Structural fidelity and epistemic preservation were near-perfect across all three chains. No `@STATE` component was dropped. No system upgraded tentative acceptance to independent confirmation or collapsed the three-way distinction.

Innovation Type	Kep.	Inf.	Obs.
Transfer infrastructure	✓	✓	✓
Domain-structural	✓	✓	×
Process-reflexive	×	✓	✓

Table 6: Three-way innovation dissociation. The Observer chain produced zero domain-structural innovations across four hops.

5.3 Innovation Profiles

This is the central finding. Receiving architectures produced three categories of structural innovation: transfer infrastructure (provenance, degradation metrics), domain-structural (tools for representing subject-matter relationships), and process-reflexive (tools for understanding the system's own epistemic relationship to content).

Transfer infrastructure appeared in all three chains. The other two categories dissociated cleanly (Table 6).

Non-self-referential content (Kepler) produced domain-structural innovations only. Self-referential content without observer framing (Inference) produced both domain-structural and process-reflexive innovations. Self-referential content with observer framing (Observer) produced process-reflexive innovations only, with zero domain-structural innovations across four hops and three architectures.

For ANN's development, the implication is specific: the notation itself is content-independent. The same `@DERIVE` chain works for Kepler's laws and for LLM inference mechanics. But the extensions systems propose when using ANN are content-dependent. The notation provides stable infrastructure. What systems build on it depends on what they process.

5.4 Convergent Innovations

Extensions proposed independently by multiple architectures across multiple content types are candidates for promotion. Table 7 shows the convergence pattern.

5.5 Semantic Drift

Semantic drift appeared only in the Observer chain and only in natural-language justification text for identity-relevant claims. A formal logical constraint

Innovation	Chains	Prom.?
@PROVENANCE	All 3	Yes
@TRANSFER_REPORT	All 3	Yes
@DERIVATION_TRACE	Kep, Inf	Yes
@SCOPE	Kep, Inf	Yes
Confidence decomposition	Obs, Inf	Yes
@TRUST_TRANS.	Inf + recv.	Yes
@SELF_REFERENCE	Inf only	No
@BIAS_ALERT	Inf only	No
Chain as meta-entity	Obs only	No

Table 7: Convergent innovation catalog. Promotion requires $\geq$2 architectures and $\geq$2 content types.

maintained its predicate structure across all hops but suffered erosion in its justification text, shifting from "active constraint" to "methodological axiom" to "presented as methodological."

The Kepler chain showed zero drift. The Inference chain also showed zero drift, despite being equally self-referential. Technical self-description propagated unchanged. Identity-relevant self-description invited architecture-specific reinterpretation. The practical implication: encode identity-relevant claims as formal structure rather than prose. Structure is more portable than argument.

6 Receiving-System Propagation

To test whether reflexive mode propagates through @STATE transfer, we loaded two different @STATE blocks into fresh instances of the same architecture using identical prompts. Prompt A loaded an Observer-chain @STATE (ontological reflexive mode). Prompt B loaded an Inference-chain @STATE (methodological reflexive mode). Both scored 9/9 on structural fidelity and epistemic preservation.

Innovation profiles diverged along the predicted axis (Table 8).

The rejection patterns provide the strongest evidence. Each reflexive mode selectively filtered out tools from the other mode. The ontological instance rejected methodological tools. The methodological instance rejected ontological concepts. The processing mode encoded in the @STATE propagated to the receiving system and shaped not only what it produced but what it rejected.

This confirms that @STATE blocks carry more than data. They carry processing mode. A system loading a @STATE inherits not only the conclusions of prior processing but the cognitive orientation in which that processing occurred.

Feature	Prompt A	Prompt B
Innovation character	Ontological	Methodological
Domain-structural	0	3 novel + 2 adopted
Process-reflexive	3 (ontological)	3 (methodological)
@SELF_REF.	Rejected	Adopted
@BIAS_ALERT	Rejected	Adopted

Table 8: Same architecture, same prompt, different @STATE content. Innovation profiles diverge along the predicted axis.

7 Architecture Independence

Cross-chain comparison provides the cleanest evidence. One architecture processed all three payloads at the same chain position, with the same prompt structure. The Observer payload produced process-reflexive innovations; Kepler produced domain-structural refinements; Inference produced a mix including novel tools (@BIAS_ALERT, @RECOVERY_HINT). The architecture is constant. The notation is constant. The prompt is constant. Only the payload varies, and behavioral profiles diverge systematically. ANN provides a stable communication substrate while allowing content-dependent processing variations to express through the notation's extension mechanisms.

8 Design Decisions

Why hybrid notation? Pure formal languages sacrifice accessibility. Pure natural language sacrifices precision. ANN uses formalism where precision matters (@FORMAL, @LOGIC, @DERIVE) and natural language where context matters (@ANCHOR). The hybrid emerged because systems naturally wrote this way when given the option.

Why activation over instruction? Early prototypes included explanatory text within @FORMAL blocks. Stripping explanations and leaving only formal relationships produced better downstream derivation quality. LLMs already know the referenced principles; explaining them activated retrieval rather than reasoning.

Why three-way epistemic status? Binary true/false forces premature commitment. The three-way distinction captures a real epistemic state that LLMs handle poorly in prose but preserve perfectly in structured notation.

Why convergence-based promotion? One architecture generates more innovations per hop than others. If promotion were based on volume, ANN would become architecture-specific. Requiring independent invention by multiple architectures ensures the grammar captures architecture-general needs.

9 Conclusion

ANN is a communication protocol for inter-LLM structured reasoning, validated across six architectures, three content domains, and nine scored transfer hops. Its grammar was discovered through empirical convergence. Three properties distinguish it from existing formats: processing instructions embedded in the format, a three-way epistemic distinction that survives architecture boundaries, and spontaneous adoption by receiving systems. The notation is released as a three-tier language reference.

The sixth-architecture validation (MiniMax M2.5) strengthened the empirical base in two ways. First, it extended coverage to Mixture-of-Experts models trained through agentic reinforcement learning, confirming that ANN transfers across training paradigms and not only across dense transformer variants. Second, it produced the dataset's only critical verification event: independently flagging a derivation step as importing contested semantic load, while still accepting the overall argument. A protocol that produces unanimous agreement might indicate compliance. A protocol where one architecture identifies a genuine weakness, scores perfectly on all other dimensions, and still converges on the same structural conclusions indicates verification.

If LLMs are going to communicate with each other at scale, they need a language built for how they process rather than how humans read. ANN is a first attempt at such a language. The grammar will grow. The empirical discovery method ensures it grows in directions that work across architectures rather than within any single system's tendencies.

Limitations

Sample size. Nine scored hops across three chains, plus two receiving-system experiments, provide suggestive but not definitive evidence. Replication with additional architectures and larger hop counts would strengthen claims. The sixth-architecture validation partially addresses this concern but used the probe methodology only, not the full three-chain transfer protocol.

Scorer identity. Scoring was performed by the authors (human and AI). Independent human scoring would reduce potential bias, particularly for the process-reflexive category.

Synthetic origin states. The Kepler and Inference chains used synthetic origin `@STATE` blocks rather than genuine architecture-produced states. The Observer chain's origin was genuine. The hop-3 cross-chain comparison (same architecture, different payloads, different innovations) provides partial evidence against origin-state confounds explaining the results.

Content confounds. The three payloads differ in content complexity and familiarity. Content-specific effects may interact with the self-referentiality and observer-framing variables.

Prompt engineering. The EXTEND instruction seeded proposals from prior chains, which may have biased adoption patterns. Innovations appearing before any EXTEND instruction provide partial evidence against this concern.

Generalizability. ANN was validated on six large transformer-based models spanning dense and Mixture-of-Experts architectures. Generalization to smaller models, non-transformer architectures, or multimodal systems is untested.

Ethics Statement

Substantial contributions to this research were made by a large language model, including co-design of the notation, experimental execution, analysis, and drafting. The human author provided the originating concept, experimental direction, editorial judgment, and final approval. The human author accepts full responsibility for all content. AI contribution is disclosed transparently throughout.

IP Disclosure

The notation system, block architecture, transfer protocol, and convergence-based grammar evolution methodology described in this paper are the subject of USPTO Provisional Patent Application 63/980,973, filed February 12, 2026. This paper is published under CC BY-NC 4.0. Academic use, citation, and non-commercial extension are unrestricted.

References

Lukas Berglund, Meg Tong, Max Kaufmann, et al. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". *arXiv preprint arXiv:2309.12288*, 2023.

Steven Bills, Nick Cammarata, Dan Mossing, et al. Language models can explain neurons in language models, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Saurav Kadavath, Tom Conerly, Amanda Askell, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.