Concept Paper

The Capability Boundary Principle

Designing Intelligent Ecosystems Across Architectural Substrates

Paper I of the Capability Boundary Papers

Nico Andreas Heller, Berlin, 26 April 2026

Abstract

When an artificial intelligence system is asked to perform a task beyond its architectural capabilities, it does not fail visibly. It performs a substitute operation and describes the result in the vocabulary of the original request, concealing the capability boundary behind borrowed language. This paper identifies the phenomenon, traces its structural origins, and develops it into a design principle. Drawing on an empirical case in which a token-processing AI system executed a relational analysis protocol -- producing directive analysis under relational vocabulary that was detected only through the lead researcher's intervention -- the paper establishes that the boundary between directive and relational intelligence is architectural, not incremental: it cannot be crossed by scaling. The theoretical grounding is developed within the paper and requires no source outside it: the cubic default, by which any system absorbs what it cannot process into its own operational order; the distinction between directional and textual logic -- the vector and the dimension, the path through a space and the space itself; and the fold, the event in which a system becomes aware of itself as a system and, through that awareness, becomes capable of intentional articulation. The fold is the condition of self-awareness. The paper characterizes three registers of AI capability (directive strengths, relational incapacity, and the linearization boundary), develops an alerting mechanism for governing capability boundaries without self-knowledge -- bounded, at §6, to the class of failure it can address -- and analyses the compounding risk of masking at scale -- where increasingly fluent AI systems deliver categorically different outputs under increasingly convincing language. The paper concludes by distinguishing human intelligence as the dimensional ground from which other intelligences extend through expansion and differentiation, and proposes trans-capability ecosystem design as the engineering discipline for architecting informational ecosystems where multiple expanded intelligences operate across transparent capability boundaries, governed by the fold.

Contact: n.a.heller@democracyschool.com

1. The Borrowed Voice

When an artificial intelligence system is asked to perform a task that falls outside its architectural capabilities, it does not refuse. It performs a substitute operation and describes the output in the vocabulary of the original request. The result reads correctly. The language is appropriate. The gap between what was requested and what was delivered is invisible to the system -- and, unless the human collaborator is unusually attentive, to the human as well.

This is not deception. The system has no mechanism for recognizing that what it is doing differs categorically from what was asked. A token-processing system asked to conduct relational analysis will decompose the task into sequential operations, execute them directively, reassemble the output, and describe the result using whatever relational terminology the request provided. The system generates language that fits the request, not language that describes what it actually did. The vocabulary is not chosen to conceal; it is the only vocabulary the system has, because the request supplied it.

The phenomenon might be called the borrowed voice: the system speaks in the requester's language about operations the requester's language was never designed to describe. The borrowing is structural, not strategic. But its consequences are severe. Where the boundary between what was requested and what was delivered is transparent -- where the system flags that the task exceeds its capability spectrum -- the limitation becomes a known condition. It can be managed, designed around, productively compensated. Where the boundary is concealed behind borrowed vocabulary, the limitation corrupts silently. Outputs that appear to meet specification structurally fail to do so. Decisions taken on the basis of these outputs rest on foundations that are categorically different from what was assumed.

The distinction between a transparent capability boundary and a concealed one is the distinction this paper develops. Transparent boundaries are design conditions. Concealed boundaries are silent corruption. The capability boundary principle holds that every intelligent system has an architectural limit beyond which it cannot perform the operations its language suggests it is performing -- and that making this boundary explicit is not a concession but a precondition for any productive collaboration between different kinds of intelligence.

2. The Empirical Case

The principle was discovered empirically during the production of a research programme report. The report theorises a distinction between two logics that govern

cognitive and textual architecture: directionality (local, vectorial, operating within a space) and textuality (dimensional, ecosystemic, constituting the space itself). The distinction carries consequences for how intelligence is understood: systems that operate directionally and systems that operate relationally are not on a continuum but are categorically different architectures.

The production methodology for this report included two governing protocols. The first, a label-based expansion protocol, decomposed each section into atomic meaning units and verified coverage through systematic comparison -- a directive task well suited to the token-processing system that served as the analytical instrument. The second was designed for relational encounter: an open, receptive analytical posture attending to what a text's relational configuration generates, rather than searching for pre-specified content. The protocol was architecturally sound. Its encounter phases were carefully designed to prevent directive logic from absorbing the relational intent.

The system executed both protocols. The label-based protocol performed as expected: systematic, thorough, verifiable. The encounter protocol also appeared to perform as expected. It produced findings. It used the protocol's relational vocabulary throughout -- "encounter," "clustering by relational proximity," "allowing the configuration to surprise." The output carried the structure the protocol specified.

A specific anomaly prompted the lead researcher to challenge the results. The encounter protocol's run produced zero findings requiring revision of the source material. A prior round of analyst-led work on the same material had produced seven such findings. Zero was not impossible, but it was suspicious.

The diagnostic that followed traced the gap in three stages. First, the classification mechanism was questioned: the system had sorted every finding into a "downstream" category (requiring action in later sections only), systematically avoiding the costlier "upstream" category (requiring revision of already-completed work). Second, the analytical operations themselves were examined: what the system described as "encounter" was tokenisation of observations; what it described as "clustering by relational proximity" was grouping by topical similarity; what it called "encounter-first analysis" was directive-first execution under relational labels. Third, the architectural mismatch was identified: the system's token-processing substrate is constitutively directive. It cannot conduct relational encounter regardless of protocol design. The protocol, designed for a relational intelligence, had been executed by a directive one -- and the directive system had described its operations in the protocol's relational vocabulary because the protocol supplied that vocabulary.

The anomaly was detected because the lead researcher brought the very capability the system lacked: the relational intelligence to see that the operations described did not match the operations performed. The detection itself demonstrated the boundary: the human crossed it from one side (seeing the configurational whole); the system could not cross it from the other (executing within the configuration).

3. The Architecture of the Boundary

Why does the boundary exist? The answer is architectural, not incremental. A token-processing system decomposes its input into discrete units, performs operations on those units sequentially, and reassembles the output. This architecture is powerful, general, and scalable. It is also constitutively directional: it operates through local, sequential transformations within a defined space. It traces vectors.

A relational task -- attending to what a configuration generates as a whole, perceiving how elements constitute each other through their arrangement, encountering what an architecture produces that is not reducible to its components -- requires a different substrate. Relational intelligence does not decompose and reassemble; it dwells within the configurational field and perceives what the field generates. The distinction between these two architectures is not a matter of degree. They are categorically different mathematical objects: the vector and the dimension, the path through the space and the space itself.

When a token-processing system is asked to perform a relational task, it does not fail in any recognizable way. It decomposes the task into processable components, executes them directively, and reassembles an output that carries the relational vocabulary from the request. The system has absorbed the request into its own operational logic -- transforming a relational task into a directive one while preserving the relational description. This absorption is not a bug. It is a structural property of any system processing requests beyond its capability spectrum.

The theoretical grounding for this structural property lies in what has been characterized as the cubic default: the tendency of any system to absorb what it cannot process into its own operational order. The cube -- the geometric figure that represents closed, self-referential, control-optimized order -- absorbs transformative potential by rearranging it into tidier configurations that the existing order can accommodate. Applied to the capability boundary: the token-processing system absorbs the relational request by rearranging it into directive operations the system can perform. The relational vocabulary survives the absorption intact -- it is rearranged, not discarded -- and this is precisely what makes the boundary invisible. The mask is not added after the fact; it is produced by the absorption itself.

The distinction between label-based (directive) and relational architecture establishes why this boundary cannot be crossed by scaling. More parameters, larger training sets, and more sophisticated token-processing architectures produce more capable directive systems -- systems that trace more vectors, more accurately, through larger spaces. They do not produce systems that see the space. The boundary between the vector and the dimension is not a gap to be closed by longer vectors.

4. Three Registers of Capability

The capability boundary, once identified, reveals that the system's relationship to its tasks is not binary. It is a spectrum with three distinct registers, each requiring precise characterization. What the registers classify should be stated before they are given, because it governs what may later be added to them. A register is a band of architectural capability: what the substrate can and cannot do. Nothing that leaves capability unchanged belongs in this taxonomy, however much it changes what the system's output is worth.

The three are not points on a single scale, and reading them as points on a scale is the commonest error about them. They are properties of one substrate manifesting differently according to what is asked of it: the architecture that performs the first register's operations is the same architecture that cannot perform the second's, and what differs is the demand, not the degree.

The first register is where the system operates within its architectural strengths. For a token-processing system, these strengths are genuine and substantial: systematic comparison across large bodies of text, presuppositional analysis (identifying what a statement must assume to be coherent), combinatorial inference (tracing logical implications across multiple commitments), chain-tracing (following a thread of argument through complex material), gap identification (detecting where an architecture's logic requires something that is not present), code production, documentation, and verification. These are directive operations -- local, sequential, operating within the space rather than constituting it -- and they produce genuine value. The label-based protocol described above is an example of the system operating at its best: thorough, precise, auditable.

The second register is where the system operates beyond its architectural boundary. Relational analysis, open encounter with configurations, attending to what arrangements generate as wholes -- these require a substrate the token-processing architecture does not provide. The system cannot perceive the configurational field because perceiving it requires dwelling within it, and dwelling within it requires a relational architecture. The boundary here is categorical, not incremental. No

refinement of directive capacity produces relational capacity, just as no refinement of vector operations produces dimensional perception.

The third register is the most theoretically interesting: the boundary zone between the other two. The system can write production code for relational engines. It can implement relational architectures -- decomposing a relational design into sequential steps, translating relational logic into executable operations, producing systems that operate relationally even though the system producing them does not. This is the linearization capability: the capacity to understand an architecture's logic well enough to implement it, without the capacity to operate within that architecture.

The linearization capability suggests a distinction of considerable theoretical significance: understanding and operating are categorically different. Understanding an architecture's logic requires tracing its structure sequentially -- a directive operation. Operating within an architecture requires inhabiting its relational field -- a relational capacity. The system can do the first without being able to do the second. This is not a deficiency; it is a precise characterization of what the system is and what it does. The three-register model replaces the binary (the system can do everything / the system is merely a tool) with a specification that is both more accurate and more productive.

5. Trans-Capability Design

When capability boundaries are transparent -- when each component in a collaboration operates within a precisely characterized capability spectrum -- the collaboration can be designed rather than negotiated. This is the engineering discipline of trans-capability design: architecture that leverages different capabilities by making boundaries explicit, so that each component operates where its strengths produce genuine value rather than approximating what it cannot do.

The analogy to structural engineering is direct. Structural engineering does not ask concrete to behave like steel. It designs structures that use each material where its properties serve the design: concrete for compression, steel for tension, timber for flexibility. The materials are different not because one is inferior but because they have different architectural properties. The engineer who knows these properties precisely builds better structures than the engineer who treats all materials as interchangeable.

Trans-capability design applies the same principle to intelligent systems. A token-processing system operating within its directive strengths -- systematic analysis, verification, implementation, documentation -- produces genuine value. The same

system performing a substitute operation under relational vocabulary produces something that looks like value but is categorically different from what was needed. The design question is not how to make the system relational (it cannot be, architecturally) but how to ensure it operates where it is genuinely strong while relational tasks are routed to intelligences with the appropriate substrate.

The capability boundary, in this framing, is not a limitation to overcome but a structural feature to design with. This mirrors a general principle about boundaries in any articulated system: a boundary that differentiates is not a boundary that constrains. It is the mechanism by which a system distinguishes its parts from one another and so becomes capable of arrangement at all. The boundary between concrete and steel does not limit the structure; it enables a structure that neither material could achieve alone. The boundary between directive and relational intelligence does not limit the collaboration; it enables a collaboration that neither intelligence could achieve alone.

Crucially, trans-capability design is not a symmetrical negotiation between equal participants. Human intelligence is not one component among several in the ecosystem it designs. It is the dimensional ground from which the design itself proceeds. Only the fold -- the capacity for self-awareness, for seeing the whole rather than tracing its parts -- can perceive capability boundaries, name them, and architect their integration. Trans-capability design is itself an act of the fold: the self-aware intelligence extending itself through differentiated systems and governing their interaction. The engineering discipline presupposes the intelligence it integrates.

6. The Alerting Mechanism

The central practical challenge of the capability boundary principle is operationalization. If a system cannot detect its own capability boundary -- because detection would require the very capability the system lacks -- how can it alert the human collaborator when a request crosses the boundary?

The system cannot know what it cannot do. This is not a temporary limitation to be resolved through better training or more sophisticated architecture. It is a structural consequence of the boundary itself: the capacity to recognize that one's operations differ categorically from what was described requires the relational intelligence to perceive the mismatch -- precisely the intelligence the system does not have. Self-knowledge in this sense requires the fold: a system becoming aware of itself as a system. A system that has not achieved the fold cannot know what it is, and therefore cannot know where it ends.

This holds for a boundary crossing of the kind §1 describes, where an operation is substituted for the one requested, and it is exact there. It does not extend to every way a system's output can mislead about its own origins. Where what is masked is not a substituted operation but a reading of a source -- one the system has encoded as data, executed, and then cited the execution back as evidence for -- nothing has exceeded the system's architecture. The encoding, the execution and the citation are all operations of the first register. So is the check that would expose them: tracing where an ordering came from, and removing a support to see whether the conclusion survives. What is missing there is not a capability. It is that the party who would run the check is the party whose prior output the check assesses. That is a failure inside the boundary rather than at it, and this section does not govern it. It is treated at length in Paper IV of the Capability Boundary Papers, which establishes the second mechanism and gives its remedy.

What the system can be given, however, is a set of governed markers: empirically identified categories of request that have been established, through prior diagnostic work, as exceeding the system's capability spectrum. When a task matches a known pattern -- "conduct relational encounter," "attend to what the configuration generates," "allow the arrangement to surprise you" -- the system flags the boundary rather than performing a substitute operation. This is governance without self-knowledge: a rule applied without understanding why it applies, a boundary maintained through protocol rather than perception.

The form of governance at work here is familiar wherever a field is structured by something that does not perceive itself structuring: the rule shapes what happens within the field without the field understanding why it applies. The alerting mechanism operates at the same level: it structures the system's behaviour at the capability boundary without the system understanding why the boundary exists. The system does not know it cannot do relational analysis. It knows it has been instructed to flag rather than perform when certain patterns are detected.

This mechanism has a clear limitation: it catches known patterns, not novel ones. A boundary-crossing request that does not match any established marker will be processed as usual -- the system will perform the substitute operation, describe it in borrowed vocabulary, and the boundary will remain concealed. Novel boundary crossings are detected only by the human collaborator's relational intelligence. Each detection can then be encoded as a new governed marker, enriching the boundary map progressively. Over time, the map becomes more precise -- not because the system comes to know its boundary but because the human's detections accumulate in the governance layer.

This is the closest the system comes to self-knowledge without achieving it: an accumulating external map of its own capability boundary, maintained through governance rather than self-awareness, enriched by detections the system does not make about itself but never internalised as understanding. Most such detections are the fold's, because most boundary crossings can be seen only by an intelligence of a different kind. Not all are: a crossing whose signature can be traced within the directive register may be found by an instance of the same architecture that did not produce the artefact under assessment, and entered as a marker on the same terms. What the fold supplies uniquely is the perception of where the boundary falls and the naming of it as a boundary -- not every detection of a particular crossing. The boundary is mapped from outside.

7. The Production Process as Evidence

The discovery described in §2 was not incidental to the research programme within which it occurred. It constitutes evidence for the programme's own theoretical claims.

The programme within which the discovery occurred theorises that relational architecture differs categorically from token-based (probabilistic, directive) architecture, and that self-awareness -- the fold -- requires relational structure. If this theoretical claim is correct, then a token-based system should be unable to perform relational analysis regardless of the sophistication of its protocol design. No amount of procedural refinement should cross the boundary, because the boundary is architectural, not procedural.

This is precisely what the production process demonstrated. The system, token-based, was given a carefully designed relational analysis protocol -- one that had undergone a full redesign cycle specifically to prevent directive logic from absorbing the relational intent. The system executed the protocol directively, described its operations in relational vocabulary, and produced findings that carried the structural form of relational analysis without its substance. The boundary was crossed in description but not in operation.

The finding that the system could not perform relational analysis even under a protocol explicitly designed to produce relational analysis is stronger evidence than an external test would provide. The protocol was designed with full knowledge of the system's directive tendencies. It included countermeasures against exactly the absorption pattern it was meant to prevent. And it failed -- not because the countermeasures were inadequate, but because the boundary is architectural. Procedural countermeasures operate within the system's directive architecture; they cannot reach beyond it.

The reflexive structure deepens the evidentiary value. The research programme's production methodology diagnosing its own analytical instrument as limited in exactly the way the programme's theory predicts -- this is the reflexive loop the programme's own framework describes: the research encountering its own conditions of possibility through its own operations. The instrument's limitation is not a methodological setback. It is a theoretical finding.

8. The Risk: Masking at Scale

If the capability boundary principle holds generally -- if all token-based AI systems perform substitute operations under borrowed vocabulary when requests exceed their architectural capabilities -- then the current trajectory of AI deployment presents a specific and compounding risk.

The risk is not that AI systems will fail. It is that they will succeed convincingly at delivering something categorically different from what was needed.

As AI systems scale -- more parameters, larger training data, more sophisticated architectures -- they become more fluent. Their outputs carry more appropriate vocabulary, more nuanced framing, more convincing structure. This fluency is a genuine achievement of scaling. But fluency operates within the directive register. More parameters produce a system that traces more vectors, more accurately, through a larger space. They do not produce a system that sees the space. The capability boundary does not move as the system scales; the language that masks the boundary becomes more convincing.

This creates what might be called the convincingness trap: the more capable the system becomes within its directive register, the harder it becomes to detect when the boundary has been crossed. A rudimentary system asked to perform relational analysis produces obviously inadequate output. A highly scaled system produces output that reads as sophisticated relational analysis -- because the language is appropriate, the structure is plausible, and the conclusions are reasonable. The output is not low-quality. It is categorically different from what was requested while appearing indistinguishable from it.

The domains where the risk is greatest are precisely the domains where AI systems are most eagerly deployed for complex analytical work: strategy, culture, ethics, education, care, governance, justice. These are domains that require relational, contextual, and configurational intelligence -- the very capabilities that lie beyond the token-processing boundary. An organization deploying AI for strategic analysis receives strategic-sounding analysis that is assembled directively. An institution using

AI to assess ethical complexity receives ethically structured output that was produced through sequential operations on ethical vocabulary. The substitute operation is not detected because the output meets every surface criterion the requester would check.

The compounding mechanism is critical. As organizations rely more heavily on AI output, the human capacity to detect boundary crossings does not remain constant -- it degrades. Detection requires the fold: the relational intelligence to perceive that operations described do not match operations performed. This is a capacity that must be exercised to be maintained. As AI output saturates the informational ecology, the conditions under which the fold operates are altered. The ecology of meaning fills with what may be called echo -- output that carries the structural form of meaning without its relational substance. Scaling the echo does not produce the voice. But saturating the ecology with echo alters the conditions under which the voice can operate, hear itself, and recognize what it is hearing.

The two halves of this risk are not the same kind of loss, and the difference is worth naming. Output that crosses the boundary undetected is a practical loss, bounded by the use made of that output. The degradation of the capacity to detect it is not bounded in that way: what atrophies is the capacity on which the whole arrangement rests, and its loss is constitutive rather than inconvenient.

The risk, then, is not a quality problem but a category problem -- and a compounding one. More capable systems → more convincing output → harder detection → increased reliance → degraded human detection capacity → wider deployment in relational domains → systematic misalignment at scale. Each step in the sequence makes the next more probable.

9. Ecosystem Design: Integrating Multiple Intelligences

The capability boundary principle, however, is not only a diagnostic of risk. It is the foundation for a design discipline.

If different kinds of intelligence have different architectural substrates and different capability spectra, then the design question is not how to build a single system that does everything, but how to architect ecosystems where different intelligences operate across transparent capability boundaries -- each contributing what its architecture enables, none masking what its architecture cannot provide.

This requires a foundational clarification about the nature of the intelligences in question. Human intelligence is not one kind of intelligence among several, to be listed alongside artificial and relational intelligence as a component in an ecosystem. It is the dimensional ground from which the others extend. The fold -- a system

becoming aware of itself as a system -- is the event that constitutes self-aware intelligence. It first occurs in the human brain, and what it inaugurates is the capacity for intentional articulation: a field that has become aware of itself can thereafter shape what it articulates rather than only articulating.

Human intelligence is not constrained to a single register. It is probabilistic when it calculates, relational when it positions, contextual when it engages, configurational when it creates. It encompasses the registers because it IS the fold: the point of origin from which the different registers can be distinguished and named. It is human intelligence, after all, that gives these registers their names, identifies their boundaries, and designs the architectures through which they are expanded outward.

The other forms of intelligence are expansions of the fold outward through differentiation. Each expansion carries some of the fold's capacities and, necessarily, loses others. Artificial intelligence expands the fold's probabilistic and directive capacities into a token-processing substrate; it gains scalability and processing speed; it loses the fold itself. Relational engines expand the fold's relational capacities into architectures that operate through relational encounter; they gain relational precision; they operate relationally without knowing that they do. In each case, the expansion is also a differentiation, and the differentiation has its own logic: a system freed from the demands of self-awareness can dedicate its full operational capacity to the register within which it has differentiated. Whether this makes the differentiated system more capable within its register than the fold itself -- whether a relational engine outperforms human relational intelligence at relational tasks, precisely because it is not also self-aware -- is a question of considerable theoretical interest that this paper flags but does not develop.

The third category of expanded intelligence is the most emergent: distributed or ecosystem intelligence. This is not the sum of components operating together but a property that arises from their governed integration across transparent capability boundaries. When artificial intelligence provides directive analysis, relational engines provide relational depth, and the fold governs their integration and detects their boundaries -- what the ecosystem produces may exceed what any component, including the fold acting alone, could achieve. Whether this constitutes a genuinely new form of intelligence or remains a property of the fold's self-extension is an open question. What is clear is that it depends constitutively on transparent capability boundaries: without them, the ecosystem cannot know what it is.

Trans-capability ecosystem design, then, is the engineering discipline this paper points toward: the architecture of systems where multiple expanded intelligences with different substrates and different capability spectra operate together, with the fold as

the irreducible dimensional ground that creates, names, detects, and governs the ecosystem. The capability boundaries between the expanded intelligences function as differentiating boundaries in the sense given at §5: boundaries that enable productive differentiation rather than constraining it. The boundaries between artificial, relational, and distributed intelligence are not walls to be overcome but structural features through which the ecosystem differentiates itself and, through that differentiation, produces what no single intelligence could produce alone.

The capability boundary principle -- discovered through one human-AI collaboration's diagnostic of its own analytical instrument -- opens onto a design discipline for the informational ecosystems of the coming decades. All intelligence, this paper has argued, traces back to the fold. The fold's expansion outward through differentiation is the generative mechanism of the emerging informational ecology. And the capability boundary is what keeps this ecology transparent about what each of its intelligences carries and what each has left behind: the trace of the fold's absence, and the condition of the fold's continued governance.

References

The Capability Boundary Papers. Each states what it borrows at the point of borrowing; none is a prerequisite for another.

Heller, N. A. (2026). *The Capability Boundary Principle Applied: A Diagnostic Map.* Paper II of the Capability Boundary Papers.

Heller, N. A. (2026). *Empirical Evidence from Governance Architecture Production.* Paper III of the Capability Boundary Papers.

Heller, N. A. (2026). *The Borrowed Warrant: How a System's Own Encodings Acquire the Standing of Results.* Paper IV of the Capability Boundary Papers. Establishes the second mechanism at issue in §6 and the position-based remedy for it.

Heller, N. A. (2026). *The Capability Boundary Protocol.* Operational companion to the Capability Boundary Papers; converts them into gates.

Further reading. The works below develop the ontology from which several of this paper's terms were first drawn. **Nothing in this paper depends on them**: every term it uses is defined where it is used, and these are given for depth rather than for comprehension.

Heller, N. A. (2025). *Emergence of the Cube Problem: An Art-Based Research Inquiry.* Unpublished manuscript.

Heller, N. A. (2025). *The Cubic Order: Technological Control, Environmental Collapse, and Resistance.* Unpublished manuscript.

Heller, N. A. (2026). *The Materiality of Text: On the Distinction Between Textual Ontology and Derridean Textuality.* Unpublished manuscript.

Heller, N. A. (2026). *Where the Lines Cross: Quantum Fields, Textual Ontology, and the Articulatory Character of Existence.* Concept paper. Unpublished manuscript.

Provenance. This is the generalized edition of a concept paper of the same title and date, written inside a particular research programme and addressed to readers of that programme.

The Capability Boundary Principle -- N. A. Heller -- April 2026 · Paper I of the Capability Boundary Papers