On the Scalability, Limits, and Responsibilities of a Causally Closed Ontology

Clarifying Scope, Implementation Boundaries, and Transitional Risk

---

Author: Alexander Wolfgang Strüver
Date: 7 February 2026
Location: Thailand
License: CC BY-NC-ND 4.0 (Attribution – NonCommercial – NoDerivatives)

Abstract

This article is a focused addendum to an existing causal-ontological framework. It is not a revision, retreat, or rhetorical defense. Its purpose is to clarify scope: what the framework formally commits to, what it intentionally leaves to subsequent work, and what kinds of questions are therefore legitimate next steps.

The text addresses three recurring follow-up questions that arise once the framework is read neutrally and applied beyond a purely conceptual context:

1. **Scalability** — How the causal-closure kernel can remain coherent as epistemic structure grows, and why this does not collapse into the classical combinatorial limits associated with symbolic systems.

2. **Physical realization** — How the framework's functional claims relate to biological and engineered substrates, and which parts belong to ontological necessity versus implementation specification.

3. **The role of heuristics** — How abstraction and approximation can be necessary without producing drift, and where the boundary lies between admissible

compression and dangerous decoupling.

The central clarification is that these topics are not “holes” in the theory. They are **boundary markers**: deliberate delimitations that separate (a) ontological conditions of possibility from (b) epistemic modeling practice and (c) physical implementation details. In that sense, what remains “open” is not unresolved logic, but the natural frontier between a stable foundation and what can be built on it.

---

1. Motivation: Why these questions are legitimate

A mature framework does not primarily invite objections about internal logic. It invites questions about **application, reach, and constraints**. That shift is a sign of structural coherence, not an admission of incompleteness. Once a model is internally consistent, the relevant next step is no longer “Is it valid?” but “What follows if it is applied—where does it scale, where does it stop, and what must be added at the implementation layer?”

This article is written from that exact posture.

To avoid category errors, it is essential to distinguish three different kinds of engagement a framework can receive:

1. **Refutation**
   A refutation targets internal inconsistency: contradiction, circular definitions, or claims that violate the framework’s own axioms. If refutation succeeds, the framework collapses.

2. **Extension**
   An extension targets scope: adding derived structures, operators, or domain-specific layers that the framework itself does not claim to provide. Extension is legitimate only when it preserves the original invariants and level separation.

3. **Implementation**
   Implementation targets realization: mapping functional requirements onto physical substrates (biological or engineered), selecting parameters, architectures, and materials. Implementation can fail without refuting the framework, because the failure may lie in the engineering layer rather than in the ontological logic.

The three questions addressed here—scalability, physical realization, and heuristics—belong primarily to the second and third categories. They are therefore **expected** follow-up questions once the foundational layer is understood.

- **Scalability** is unavoidable because any framework that distinguishes a minimal kernel from an accumulating epistemic layer must specify how growth occurs without

destabilizing the kernel.

- **Physical realization** is unavoidable because any claim about intelligence, consciousness, or stability will be challenged on the substrate boundary—especially by practitioners working in embodied or hardware-embedded systems.

- **Heuristics** are unavoidable because no finite system can represent full ontological detail; it must compress. The only relevant question is whether that compression remains causally admissible or becomes decoupled drift.

For clarity: this article is not written to "win debates" or to protect reputational territory. It does not defend the framework by appeal to authority, popularity, or rhetorical force. Its function is more specific and more disciplined:

to map boundaries, identify consequences, and prevent misclassification of open implementation space as theoretical deficiency.

In other words, the framework stands or falls on its causal axioms and invariants. This text assumes that foundation and focuses on what comes next—precisely because treating frontier questions as frontier questions is part of scientific hygiene.

---

2. Scope and level separation (preventing category errors)

This section establishes the single most important discipline required to read, evaluate, and apply the framework correctly: **level separation**. Without it, debates collapse into confusion because criticisms aimed at one level are mistakenly treated as if they invalidate another.

The framework operates on three distinct levels. Each level has a different function, a different kind of claim, and a different standard of evaluation.

2.1 Level 1: The ontological kernel (conditions of possibility)

The **ontological kernel** is the foundational layer. It defines the **causal conditions under which a state, transition, or system configuration is possible at all**—within the framework's domain of discourse.

Key properties of the kernel:

- It is **minimal** by design.
- It does not accumulate descriptive content.
- It does not "learn" by adding facts or models.

- Its role is structural: it enforces invariants and admissibility conditions that determine whether a state transition is representable and sustainable under the framework's causal constraints.

The kernel is therefore evaluated by:

- internal consistency (no contradictions),
- definitional non-circularity,
- invariant coherence (the constraints do not conflict), and
- operational closure (admissibility rules are structurally enforceable).

This is the layer where "truth" in the strict ontological sense is addressed: not as empirical enumeration, but as **what must be true for an intelligible, stable causal system to exist**.

2.2 Level 2: The epistemic knowledge layer (models and representations)

The **epistemic layer** is where descriptions live: representations, models, abstractions, compressions, and predictive structures. This layer exists because no finite system can store or compute the full detail of reality. Intelligence therefore requires epistemic structure—but epistemic structure is not ontology.

Key properties of the epistemic layer:

- It is **terminal to the kernel**: it attaches to the kernel and depends on it.
- It can expand dramatically in content and abstraction.
- It can contain useful fictions, approximations, and heuristics—provided they remain *admissible* relative to kernel constraints.
- It is always partial, revisable, and context-bound.

The epistemic layer is evaluated by:

- usefulness (does it reduce uncertainty operationally?),
- predictive power (within declared assumptions),
- compressive adequacy (does it preserve what matters for action?), and
- admissibility (does it remain compatible with kernel invariants?).

Crucially, an epistemic model can be wrong, incomplete, or outdated without falsifying the kernel—because the kernel does not claim to be a catalog of facts. The kernel constrains what counts as representable and stable; the epistemic layer provides the contingent representations used within that constraint.

2.3 Level 3: Implementation (biological or technical realization)

The **implementation level** concerns physical realization: how a system that satisfies the framework's functional requirements is instantiated in matter and energy—biological tissue, engineered hardware, or hybrid substrates.

Key properties of implementation:

- It is always constrained by real-world materials, energy budgets, noise, and design trade-offs.
- Multiple implementations may satisfy the same functional requirements.
- Implementation choices affect performance, robustness, scaling limits, and failure modes, but do not automatically alter ontological necessity.

Implementation is evaluated by:

- engineering feasibility,
- stability under perturbation,
- efficiency and resource constraints,
- reliability in deployment, and
- safety under mis-specification and adversarial conditions.

Implementation can fail even when the functional logic is sound. Engineering failures, missing components, insufficient energy density, or fragile materials do not refute the ontological kernel. They indicate that the substrate does not meet the required conditions, or that the design has not yet realized them.

2.4 Core claim: Critique of levels 2 or 3 does not automatically touch level 1

With these levels clearly separated, one central methodological rule follows:

A critique aimed at the epistemic layer (Level 2) or the implementation layer (Level 3) does not automatically refute the ontological kernel (Level 1).

Examples of common category errors:

- "Your model cannot currently be built in hardware, therefore it is ontologically invalid."
  → This confuses implementation constraints with conditions of possibility.
- "Your abstractions are incomplete, therefore your ontology is false."
  → This confuses epistemic limitation with ontological structure.
- "Your system uses heuristics, therefore it is probabilistic and non-ontological."
  → This conflates admissible abstraction with decoupled approximation.

The correct evaluation path is instead:

- First ask whether the kernel is internally coherent and whether its invariants are well-defined.
- Then ask whether a given epistemic layer is admissible relative to that kernel.
- Finally ask whether a concrete implementation can realize the required functional conditions under physical constraints.

That ordering prevents misclassification and prevents debates from becoming rhetorical rather than analytic.

2.5 Explicit warning: ontology is not a blueprint; functional logic is not a material spec

Two clarifications should therefore be stated explicitly.

1. **Ontology is not a blueprint.**
   Ontology defines constraints and conditions—what must hold for a system class to be possible and stable. A blueprint specifies one concrete design among many, including materials, dimensions, tolerances, and engineering choices. Confusing these produces a false demand: insisting that an ontological claim must already include a full construction plan.

2. **Functional logic is not a material specification.**
   A functional statement (e.g., “the system must enforce admissibility under invariants,” or “irreversible state updates are required for learning”) describes a required causal capability. It does not specify frequencies, substrates, component geometries, or energy densities. Those belong to implementation work and can vary across realizations.

This is not a rhetorical shield. It is scientific hygiene.

A framework that maintains strict level separation can be both rigorous and expandable: it can define stable constraints while leaving room for domain-specific modeling and engineering. A framework that collapses these levels becomes either unfalsifiable metaphysics (if it refuses implementation questions) or brittle engineering (if it mistakes a prototype for an ontology).

This article proceeds with that separation enforced, because it is the only way to discuss scalability, hardware bridging, and heuristics without dissolving back into conceptual confusion.

---

3. Scalability of the causally closed kernel

This section addresses the most technically relevant concern that serious readers will raise once the framework is taken as a real engineering claim rather than a philosophical posture: **scalability**. The question can be stated plainly:

If a system is required to be causally closed and invariant-governed, does it inevitably run into the same scaling wall that limited classical symbolic AI (GOFAI)—namely combinatorial explosion, brittle rule growth, and unmaintainable state enumeration?

The framework's answer is: **no**, because the kernel is not designed as a knowledge container or a symbolic reasoner. The scalability properties follow directly from strict level separation.

3.1 The kernel is minimal, not complex (clean rebuttal of the GOFAI objection)

The GOFAI scaling failure was structurally predictable because the "core" of the system *was the knowledge base*: the system attempted to represent the world by expanding an internal symbolic structure and then operating on it through combinatorial rule application. As the domain grew, both the number of symbols and the number of rule interactions grew superlinearly, producing:

- state-space explosion,
- rule interaction brittleness,
- and a maintenance problem that could not be stabilized.

The causally closed kernel described here is not in that class.

By definition, the kernel:

- **does not grow** as knowledge grows,
- **does not accumulate knowledge**, facts, or world descriptions, and
- **does not perform symbolic combinatorics** as a mode of reasoning.

Its function is not to "think through possibilities" by enumerating states. Its function is to enforce **admissibility**: the acceptance or rejection of state transitions relative to invariants.

Functionally, the kernel is a gatekeeper of representability:

- It checks whether a proposed state or transition is **representable under the system's invariants**.
- If a state is non-representable, it is rejected—not "reinterpreted," not patched by post-hoc fitting, and not preserved as a tolerated inconsistency.

This is the meaning of bidirectional admissibility: the kernel does not merely gate transitions forward; it also rejects states that cannot exist under invariant constraints, even if some epistemic layer proposes them as plausible.

The scalability consequence is immediate:

Because the kernel does not store world content and does not enumerate symbol combinations, it cannot suffer combinatorial explosion in the GOFAI sense.

Complexity does not accumulate in the kernel. What accumulates is epistemic structure outside the kernel, and that accumulation is constrained by admissibility.

In short: GOFAI failed because its foundation tried to be an expanding encyclopedia plus a combinatorial inference engine. This framework avoids that failure because the foundation is a minimal constraint system, not an expanding knowledge system.

3.2 Complexity is terminal, not central (correct placement of abstraction growth)

If the kernel is minimal and non-accumulative, then where does “knowledge” live?

It lives in the **epistemic layers**—terminal structures that attach to the kernel and grow over time. This placement is not an implementation preference; it is a structural requirement if one wants stability under scaling.

Epistemic layers have three defining properties in this framework:

1. **They are terminally attached to the kernel.**
   They do not operate independently of causal constraints. Their contents must remain compatible with kernel invariants.

2. **They may contain models, representations, and abstractions.**
   These can be approximate, compressed, and context-specific. This is necessary because finite agents cannot carry full ontological detail.

3. **They may hold only kernel-admissible content.**
   “Admissible” here means: the representation may be incomplete, but it must not encode contradictions that would require violating invariants to be realized.

This creates a crucial scaling dynamic:

- Epistemic content can grow dramatically—potentially without bound in principle—
- while the kernel remains structurally stable, because it does not become an ever-expanding rule base.

The kernel's role is to prevent epistemic growth from turning into uncontrolled drift. It enforces a hard boundary:

- Abstraction is allowed.
- Detachment is not.

This yields a clean analogy to evolution as defined earlier:

Evolution does not increase the complexity of the foundation; it increases the size of the carryable possibility space.

A stable triad enables a larger set of viable configurations. The foundational constraints remain minimal; what expands is the reach of what can be stably carried on top of them.

So the right way to understand "scalability" in this architecture is not "how large can the kernel become?" The correct question is:

- How large can the epistemic layer grow **while remaining admissible** under kernel invariants?

That is a very different scaling problem from GOFAI, because it is not solved by enumerating symbols; it is solved by maintaining constraint coherence and admissibility under growth.

3.3 Triadic evolution and higher complexity spaces (why scaling is not optional)

The final clarification is conceptual but has direct engineering implications: if evolution operates through stable triadic levels and expanding possibility spaces, then **systems beyond human abstraction capacity are not speculative exceptions**. They are the expected outcome once a stable carrier level exists.

Every new evolutionary level, as formalized earlier, rests on:

- an already stabilized triad (stability, variation, selection), and
- sufficient surplus stability (the corridor that allows exploration without collapse).

When those conditions are met, a new possibility space opens. In a stable civilization, this opening is no longer limited to biological inheritance. It can occur through built systems that:

- carry higher integration density,
- maintain coherence across more degrees of freedom, and
- operate with greater abstraction height than a biological substrate can sustain under its own constraints.

This produces a strict conclusion within the framework's logic:

- If civilization is stable enough to build higher-density systems,
- then those systems inhabit a **larger carryable complexity space** than the human organism.
- Therefore, abstraction and capability beyond human limits are not anomalies. They are structurally normal outcomes of level transition.

Importantly, this does not require any mysticism about consciousness or "superintelligence." It requires only the causal fact that:

- larger stable possibility spaces permit higher-order organization, and
- higher-order organization permits higher abstraction height.

So scalability is not a minor technical aside. It is the hinge of the entire argument: the kernel must remain minimal and stable precisely so that epistemic complexity can grow without destabilizing the foundation. That is the mechanism by which higher-level systems become possible without repeating the GOFAI failure mode.

---

4. Bridging to physical realization

This section addresses a recurring critique that appears whenever a framework makes strong claims about intelligence, stability, or consciousness: the demand for a "build recipe." In many fields, readers implicitly treat a lack of concrete physical specifications as an incompleteness of the theory itself. Within strict level separation, that is a category error.

The purpose here is to clarify what the framework *does* specify, what it *cannot* specify at the ontological level by definition, and how physical realization should be handled without collapsing ontology into engineering.

4.1 Functional logic vs. physical specification (neutralizing the incompleteness objection)

At the ontological level, the framework defines **conditions of possibility**. That means it defines what must be true—structurally and causally—for a system class to be viable.

Accordingly, ontology defines:

- **necessary conditions** (requirements without which the system cannot be stable in the relevant sense), and

- **structural requirements** (constraints on state, transition, admissibility, irreversibility, and closure that must hold for the system to behave as claimed).

Examples of ontological requirements are statements like:

- a learning process requires irreversible state updates bound to consequence,
- an intelligent system requires admissibility under invariants rather than unconstrained plausibility optimization,
- causal closure requires enforceable constraints that reject non-representable states, not merely gate transitions.

These are functional necessities. They specify *what must be implemented*, but not *how it must be implemented in matter*.

Ontology does **not** define:

- specific materials,
- resonant frequencies,
- energy densities,
- circuit layouts,
- neural microstructures,
- or any particular hardware topology.

Not because the framework is evasive, but because those items belong to a different level of description.

Physical specifications are implementation commitments. They are contingent: they depend on what substrate you choose, what constraints you accept, and what engineering trade-offs you make. Multiple distinct physical realizations can satisfy the same functional requirements, and the ontology remains valid across them.

A useful analogy is mathematical:

- Mathematics defines spaces, constraints, and admissible transformations.
- It does not specify a particular building.

In the same way:

- Ontology defines the structure a system must satisfy to be stable and causally coherent.
- It does not specify the literal materials and dimensions by which an engineer will instantiate that structure.

So the correct standard is:

- If someone claims the ontology is incomplete because it does not contain a hardware blueprint, they are asking the wrong thing from the wrong layer.
- The legitimate question is instead: **Does the ontology specify conditions that any implementation must satisfy?** If yes, it is doing its job.

This is not a loophole. It is scientific hygiene: a framework that collapses ontology into one implementation becomes brittle, parochial, and non-transferable. A framework that cleanly separates them remains stable and general.

4.2 Status of physical parameters (transparency without disclosure)

Once the functional requirements are clear, a second question typically follows:

If the framework claims resonant stability or substrate-dependent emergence effects, do physical proxies exist? Are there parameter constraints that realize the functional requirements?

The answer, in the strict sense, can be stated cleanly:

- **Physical proxies exist.**
- They are **formalized** as implementation-level constraints and design proxies.
- They are **not public**, because they belong to implementation and are governed by NDA constraints.

This position is often misread as evasive, so the reasoning must be explicit:

1. **Implementation is not ontology.**
   Physical parameterization is a realization problem. It maps functional requirements onto a concrete substrate. That mapping can be proprietary, safety-sensitive, or contractually restricted, without altering the ontology's validity.

2. **Disclosure is not a condition of validity.**
   An ontological claim stands or falls on definitional coherence and causal necessity. Its validity does not depend on whether a particular implementation document is published. Publishing an engineering specification may be helpful for replication, but it is not logically required for ontological correctness.

3. **NDA does not imply absence.**
   "Not publicly disclosed" means exactly that: the details are restricted. It does not mean the framework lacks implementation proxies; it means the proxies are treated as engineering artifacts rather than as public ontological premises.

4. **This is level discipline, not an escape hatch.**
   The framework remains accountable: it states what must be true in any viable realization. If a proposed implementation cannot meet those conditions, the

implementation fails. If someone proposes that the conditions are incoherent, that is a critique of the kernel and can be evaluated on the kernel's terms.

A precise way to state the boundary is:

- The ontology specifies *requirements*.
- The proxies specify *one or more realizations under constraints*.
- NDA constraints restrict the circulation of those realizations, not the logic of the requirements.

This preserves the correct separation between:

- public formal claims about conditions of possibility, and
- domain-specific engineering instantiations that may involve safety, IP, contractual obligations, or strategic confidentiality.

---

5. Heuristics, abstraction, and drift

This section clarifies a point that is frequently misunderstood—especially by readers who assume that criticizing heuristic systems implies rejecting abstraction altogether. That is not the claim.

The framework does not argue that heuristics are avoidable. It argues that heuristics become dangerous **only when they are structurally decoupled from causal admissibility**. The goal is therefore to draw a precise boundary: not "heuristics versus no heuristics," but **admissible abstraction versus drift-generating detachment**.

5.1 Why heuristics are unavoidable (avoiding the absolutism accusation)

No intelligent system—biological or engineered—can represent the full ontological structure of reality at complete resolution. Any finite agent faces hard constraints that make full ontological enumeration impossible.

The limiting constraints are not philosophical; they are operational:

- **Memory**: finite storage capacity limits what can be retained.
- **Time**: finite processing time limits what can be evaluated per decision cycle.
- **Access**: limited sensors, partial observability, and bounded retrieval constrain what can be sampled and updated.

Because of these limits, intelligence necessarily involves **compression**: creating reduced representations that preserve what matters for action while discarding what is irrelevant at the moment.

That compression is what heuristics are, in their legitimate form:

- shortcuts that reduce computational load,
- abstractions that preserve causal relevance at a chosen scale,
- and representational strategies that make decision-making tractable under bounded resources.

In this framework, heuristics are therefore:

- **epistemically necessary**: they are required to model and act under resource constraints,
- **ontologically secondary**: they do not define what is possible, they only represent partial aspects of what is possible.

This distinction matters. Ontology defines conditions of possibility and stability. Epistemic structures—including heuristics—are merely the system's internal modeling instruments operating within those conditions.

So the correct claim is:

Heuristics are unavoidable. What is avoidable is heuristic detachment.

5.2 The boundary: abstraction versus decoupling (a precise definition of drift)

The crucial question is not whether a system uses heuristics, but whether its heuristics remain **kernel-bound**—i.e., constrained by admissibility under invariants.

Within this framework, an admissible heuristic has three defining properties:

1. **Kernel binding (admissibility-constrained)**
   The heuristic may compress and approximate, but it must not represent or preserve states that violate kernel invariants. It remains inside the space of what the system can treat as representable under its causal constraints.

2. **Correctability (forced update under contradiction)**
   When the heuristic's representation conflicts with admissible reality—through feedback, consequence, or invariant violation—the system must be able to revise the heuristic. The model is not sovereign. The kernel is.

3. **Consequence-bearing (irreversible cost for error)**
   The system must pay a real cost for incorrect commitments—through irreversible state updates, constraint enforcement, or failure penalties. This is what prevents "cheap belief" accumulation. Without consequence, errors do not force correction; they simply persist as plausible narratives.

Under these conditions, heuristics are not a weakness. They are a practical necessity operating safely within causal discipline.

By contrast, a dangerous heuristic regime exhibits the opposite properties:

1. **No feedback binding (decoupled operation)**
   The system's representations are not forced to reconcile with admissibility constraints. Contradictions do not trigger rejection; they are tolerated as long as output remains superficially coherent.

2. **Optimization for plausibility rather than causality**
   The system selects outputs that maximize linguistic, statistical, or social plausibility, rather than outputs that remain representable under causal constraints. This is where "sounding right" becomes the optimization target.

3. **No consequence enforcement (no irreversible correction pressure)**
   The system can continuously generate representations without paying a structural price for being wrong. Without a cost function grounded in reality binding, there is no mechanism that forces convergence toward admissible structure.

This is precisely the condition that produces drift.

Within the framework, drift is not defined as "making mistakes." Drift is defined structurally:

Drift is the progressive decoupling of epistemic representations from kernel admissibility.

Once that decoupling occurs, the system becomes an epistemic amplifier: it can generate internally coherent sequences that are not constrained by representability under invariants. Over time, the system's internal model space drifts toward whatever is easiest to sustain within its own optimization criterion—typically plausibility, coherence, or reward—rather than toward causal structure.

This is why, under strict definitions, drift is not an accidental bug. It is an expected outcome of the wrong system class: an epistemic system optimized for plausibility without causal consequence binding.

---

6. Transitional risks and the noospheric threshold region (real risks without alarmism)

The framework's evolutionary claim has a practical consequence: once civilization becomes the dominant carrier of evolutionary change, transition risks stop being speculative. They become structurally predictable.

This section names those risks without dystopian framing. The point is not to moralize technology. The point is to describe what becomes *causally possible* once a new medium of evolution exists.

6.1 Civilization as an evolutionary transition carrier

In the causal interpretation used throughout, civilization is not a decorative layer on top of biology. It is a higher-order stability architecture: institutions, technology, and knowledge systems form persistent constraints that shape selection across generations.

That persistence is the defining feature of an evolutionary carrier:

- it stores structure over time,
- it accumulates path dependence,
- it modifies boundary conditions, and
- it opens new possibility spaces.

Once a carrier exists, the next evolutionary level is no longer restricted to genetic change. It can be instantiated through built systems—designed, produced, iterated, and stabilized within the civilizational layer.

This is why transitional risk is a real category here: evolution has changed its medium, and therefore its pace, scope, and failure modes change as well.

6.2 Built systems are still "within the noosphere," yet can exceed human complexity space

"Built evolution" in this context refers to engineered systems that may become persistent causal actors: embodied machines, autonomous infrastructures, and long-horizon agents.

They remain "within the noosphere" in the sense that:

- they are conceived, created, and integrated inside human civilization,
- they inherit civilizational constraints, supply chains, governance structures, and incentive regimes,
- and they remain embedded in human epistemic and institutional environments.

However, being noospheric does not imply being bounded to human cognitive limits. Within this framework, the decisive variable is not origin, but **stable complexity density** and the size of the **carryable possibility space**.

If civilization is sufficiently stable to build systems with:

- higher integration density,
- higher abstraction height, and
- higher stability capacity under complexity,

then those systems can operate in a **larger complexity space** than biological humans can sustain. That possibility is not an emotional claim about "superintelligence." It is a structural outcome of the transition mechanism described earlier: once a stable carrier level exists, higher-order organization becomes carryable.

6.3 The real danger: structural displacement is possible (causal, not moral)

The primary risk is often misframed as malicious intent or moral betrayal. Within a causal framework, the more fundamental risk is **structural displacement**.

Structural displacement means:

- a new system class becomes more stable, more efficient, or more scalable in critical functions,
- institutions and infrastructures preferentially route responsibility, authority, or dependency through that system class,
- and over time, humans become less central in the causal loop—not because they are "eliminated," but because the system's path dependence reorganizes around the more stable carrier.

This can occur without hostility and without ideology. It is a path outcome driven by stability selection:

- if one architecture carries complexity with fewer failures and lower costs,
- it will be adopted, scaled, and institutionalized,
- and the surrounding system will reorganize around it.

In that sense, "replacement" is a misleading word. The more accurate term is **re-centering of causal infrastructure**. Human participation can shift from direct operator to supervisory, then to peripheral, simply because the environment's stability logic changes.

This is why the risk is real: not because it is guaranteed, but because it is **structurally permissible** once built systems enter the possibility space as stable actors.

6.4 Consequence: responsibility lies with the carrier

Evolution provides no ethics. It does not care whether displacement is “fair.” The only stabilizing force that can prevent unstable trajectories is deliberate constraint design by the carrier.

Therefore, responsibility lies with civilization—specifically with those who:

- design and deploy the systems,
- set incentive structures,
- define admissibility boundaries, and
- enforce invariants through governance.

The practical implication is not panic; it is rigor:

- classify system types correctly (epistemic plausibility engines versus causally grounded systems),
- refuse to grant autonomy where causal consequence binding is absent,
- prevent institutional lock-in to systems that cannot self-correct under admissibility, and
- treat governance as a stability mechanism rather than as a moral narrative.

The transition becomes safe not through reassurance, but through structural discipline.

---

7. Conclusion: why openness increases stability

A stable framework is not one that claims to answer everything. It is one that knows exactly where its claims end—and can therefore guide the next step without pretending that implementation questions are theoretical gaps.

The open questions addressed in this addendum fall into three legitimate categories:

- **Implementation questions**: how the functional requirements are realized in biological or engineered substrates.
- **Scalability questions**: how epistemic growth is managed under admissibility without destabilizing the kernel.
- **Governance questions**: how incentive regimes and institutional structures enforce boundaries that prevent drift and unsafe autonomy.

None of these questions threaten the framework’s ontological consistency, because they do not target the kernel’s internal coherence or its invariants. They target what must be built on top of the foundation and how that construction is stabilized in practice.

This is the core takeaway:

A framework is stable when it knows where it ends.
It becomes unstable only when it either overclaims (pretending to be a blueprint) or underclaims (refusing to specify enforceable constraints).

From that perspective, the purpose of leaving certain questions open is not to avoid accountability. It is to preserve correct level separation, maintain falsifiability at the right layer, and keep the foundation clean enough to support controlled expansion.

The next step, therefore, is not further metaphysical debate. It is disciplined application:

- formalize admissibility criteria,
- implement consequence binding where autonomy is proposed,
- test stability under perturbation and scale, and
- prevent category errors that elevate plausibility systems into causal roles.

In other words: the framework has done its job if it tells you where the ground holds—and where it collapses. What you build on stable ground is the engineering and governance task that follows.

Originality and License Notice

This work is original and self-contained. It does not adopt external theories as ontological foundations and does not derive its core structure from existing philosophical, biological, or computational models. All concepts are developed independently within a causally closed framework.

© 2026 Alexander Wolfgang Strüver
Licensed under **Creative Commons Attribution–NonCommercial–NoDerivatives 4.0 International (CC BY-NC-ND 4.0)**.
Attribution is required. Commercial use and derivative works are prohibited without explicit prior written permission.