# Highways and Train Tracks

## Why Genuine Determinism in AI Systems Requires a Different Architecture — and What the Industry's "Deterministic Guardrails" Actually Mean

*A Kindynos Advisory Whitepaper*

Santa Federico
Founder & Principal, Kindynos Advisory
29 May 2026

*Research analysis and strategic commentary. Not legal, tax, investment, accounting, or regulatory advice.*

## Contents

## Preface — scope and argument

The artificial-intelligence industry has converged in the past eighteen months on a specific vocabulary for the problem of unreliable model output: hallucination, guardrails, alignment, reliability, deterministic infrastructure. The vocabulary is useful but conceals a more important architectural distinction that the leading platforms tend not to draw clearly — the distinction between making a probabilistic system *less likely* to deviate, and building a system that *cannot* deviate by construction.

This whitepaper draws that distinction sharply, locates it in the academic and industrial literature, and argues that the second category — what the academic field calls **neurosymbolic** or **hybrid** AI, and what practitioners increasingly call a **symbolic core with the language model at the edges** — is a fundamentally different architectural class from a large language model wrapped in guardrails, however sophisticated those guardrails become. The paper presents the analogy that motivates the distinction (highway-with-fences versus train-on-tracks), surveys what major AI platforms actually deliver under the label "deterministic" or "reliable," reviews the academic literature on neurosymbolic and formally-verified hybrid systems, presents a worked example of a purpose-built deterministic application, and states the trade-off honestly: a system on tracks can only go where the tracks lead, and that is not a limitation to be apologized for but the property that gives it its credibility.

> *The argument is not that probabilistic AI is poor engineering — it is the right architecture for many problems. The argument is that for certain classes of work, particularly in regulated finance, in risk management, in safety-critical operations, in any domain where a deviation is not "a stylistic issue to be patched in the next iteration" but "an outcome the system was supposed to make impossible," the probabilistic-plus-guardrails architecture is structurally insufficient. The honest answer in those domains is not better guardrails; it is a different kind of system.*

## 1. The distinction

The clearest way to see the architectural distinction is by analogy.

A **highway with guardrails** is a transportation system in which a vehicle has a near-infinite range of possible trajectories, and the engineering effort is directed at *constraining* those trajectories: lane markings, rumble strips, guardrails, crash barriers, signage, speed limits, traffic enforcement. The system is enormously flexible — cars can go almost anywhere, follow many possible routes between two points, deviate for any number of legitimate reasons — and the engineering goal is to make *unwanted* deviations sufficiently rare or sufficiently contained.

A **train on tracks** is a transportation system in which the vehicle has, by mechanical construction, exactly one possible trajectory at any moment: forward along the track

or backward along it. The engineering effort is directed not at constraining deviation but at *eliminating its possibility*. A train cannot leave the track without catastrophic failure; it cannot lane-change; it cannot decide to take a scenic route. The track determines where it goes. The trade-off is severe: trains can only travel where someone has already laid track. They are useless for door-to-door transport, for novel destinations, for problems that require flexibility about route. But on the routes the tracks cover, the operational properties are different in kind, not in degree: precise scheduling, high throughput, no driver judgment required, no possibility of departure from the planned trajectory.

The analogy is not loose. Software engineering has the same two categories. A traditional, rule-based, deterministic program is a train on tracks: given an input, it can produce exactly one output, the one the rules specify; there is no possibility of variation, no creativity, no graceful handling of inputs the programmer did not anticipate. A modern large language model is a highway with guardrails: given an input, it produces a distribution over possible outputs, and the engineering work is to bias that distribution toward acceptable answers and to catch unacceptable ones before they reach the user.

The industry's term for that second mode of engineering — bias the distribution, catch the bad ones — is **guardrails**. The term is appropriate and the analogy holds: guardrails are the engineering response to a system that can, in principle, go in many directions. They are not, however, what a train provides. A train does not have guardrails because it does not need them. The tracks *are* the constraint, not an addition to the constraint.

> *The conflation matters because in the industry's current vocabulary, "deterministic guardrails" and "deterministic infrastructure" appear in nearly every enterprise-AI vendor's marketing, and they are used to mean both things — sometimes a system bounded by sufficient checks that its outputs are predictable enough for production use, and sometimes a system that cannot deviate by construction. These are different architectures and they have different operational properties. A buyer who is told they are buying determinism, and who needs the second meaning, is being sold the first.*

## 2. What the industry actually delivers under "deterministic"

The convergence on the language of determinism in 2025 and 2026 is real and traceable. Production engineering blogs, vendor marketing, and academic-industrial collaborations all use the term. But examined closely, almost every published reference to "deterministic" infrastructure for AI describes the guardrails category, not the tracks category. This is not duplicity; it is the natural state of the field. Building genuine deterministic systems is harder than wrapping a probabilistic one with checks, and the market reward for the wrapped version is faster.

A representative summary, from a recent technical analysis of deterministic guardrail stacks: "The key idea is simple: reliability should not depend on the model

behaving well. It should depend on the system enforcing correctness." The architecture proposed in that article is a layered enforcement stack — design-time prompt linting, CI-time contract checks, runtime structural and evidential verification. Each layer adds determinism around the model. The model itself remains probabilistic. The reliability emerges from the architecture, not from the model.

Another technical guide is candid about the fundamental constraint: "LLMs are inherently never deterministic, but there are a surprising number of tricks you can use to build a system where you can fully trust the output to be what you expect." The framing — *tricks* to make a non-deterministic system trustable — is precise. The model is not deterministic; the surrounding engineering is.

The most-cited practical pattern in this literature is the layered guardrail architecture. A widely-circulated production-AI article describes it: "Hard-coded rules that override the LLM. Fast, deterministic, reliable. 'If output contains a medication dosage recommendation, block.' 'If output suggests a specific diagnosis without citing the source transcript, block.' 'If output contains language that could be interpreted as medical advice to a patient, rewrite the framing.' These are your absolute safety boundaries. They don't need to be smart — they need to be fast and never wrong… The limitation: rules only catch what you've anticipated. They miss novel phrasings, edge cases, and subtle violations."

This is a clear, honest account of how the guardrails architecture works in practice. It is also a clear, honest account of why it is not genuine determinism. The architecture explicitly acknowledges that the rules only catch what has been anticipated; the layers above the rules — LLM-as-judge, schema validators, clinical validators — are themselves a mix of deterministic and probabilistic components. The system as a whole has a reliability that is high but not absolute, with the unhandled cases routed to human review. That is good engineering for many problems. It is not what a train provides.

A vendor analysis from early 2026 is more direct about the architectural shift now under way: "Many workflows need deterministic formatting, consistent language, and traceability. Privacy, consent, and data residency. Regulated industries require strict control over what data is processed, where it goes, and how it's logged and retained. That's where domain-specific models and hybrid architectures shine." The same source articulates the design pattern that the field has, in practice, settled on for high-stakes work: "keep the rules-based core where correctness is critical, and use the LLM where it creates the most value — interpretation, explanation, and speed of development."

> *This is the architectural insight that the marketing language often obscures: in mature deployments in regulated industries, the core is deterministic — built out of traditional rules-based software, hard constraints, and where possible formal verification — and the LLM is at the edges, doing the things the deterministic core cannot do well (interpreting unstructured input, summarizing for human consumption,*

> *explaining results in natural language). The reliability of the system comes from the core, not from the model.*

A representative source on the broader picture states it as a design principle (Zartis, January 2026): "In production, [randomness] fuels bugs. Business applications, from compliance automation to customer support, need models that act like trustworthy teammates, not poets improvising under pressure. True innovation in LLM engineering will come not from discovering more surprising model behaviors, but from mastering predictable performance." The diagnosis is correct, but the proposed remedy — "engineering determinism" through better infrastructure around the LLM — is the highway-with-guardrails answer. It is one valid answer. It is not the only architectural option.

## 3. The academic picture — neurosymbolic AI and the symbolic core

The architectural option that does deliver genuine determinism in the train-on-tracks sense has a name in the academic literature: **neurosymbolic AI** (sometimes called hybrid AI, sometimes formal-verifier-in-the-loop). The defining property of these systems is that an output produced by a neural component is not accepted until it has been validated by a separate, deterministic symbolic component, and the symbolic component has formal correctness guarantees within its domain of competence.

A clarification is worth making early, because the literature often blurs it: determinism is not the same as formal verification. Deterministic software guarantees reproducibility — the same input state yields the same output state. Formal verification, where it is available, additionally proves specified properties of that system or its transitions. The strongest architectures combine both, but many useful deterministic-core systems are not formally verified end to end; the claim of this paper is the reproducibility-and-inspectability property, with formal proof as a further layer where the substrate supports it.

Kognitos, a vendor in neurosymbolic automation, articulates the principle directly: "Neurosymbolic AI combines neural networks for perception and understanding with symbolic reasoning for deterministic, verifiable execution — eliminating the hallucination risk inherent in purely generative AI systems. Pure LLMs predict the most statistically likely output; neurosymbolic AI validates every output against formal business rules before execution, making it the only architecture suitable for mission-critical automation."

The mechanism is what matters. In a guardrails architecture, the model produces a candidate output and the guardrails check it for compliance with rules that have been written in advance to catch known failure modes. In a neurosymbolic architecture, the symbolic component does not check the model's output for compliance — it produces the actual operational output itself, with the model serving as a *suggestion engine* whose suggestions are then either confirmed by symbolic

reasoning or discarded. The difference is whether the deterministic component is downstream (filtering) or in the critical path (deciding).

The academic literature contains worked examples that make this concrete. The **SymCode** framework (arXiv 2510.25975v2; EACL 2026 Findings) reframes mathematical problem-solving as verifiable code generation: an LLM generates a Python script, the script is executed in a sandboxed interpreter that provides a deterministic pass/fail signal, and execution errors are fed back to the LLM for self-correction. The reliability of the system is not assured by trusting the LLM's prose reasoning; it is assured by the deterministic interpreter, which either runs the code and gets the right answer or doesn't. The researchers report up to 13.6 percentage-point accuracy gains over prose-based prompting on benchmarks including MATH-500 and OlympiadBench, with the gap widening as problem difficulty rises. The lesson is precisely the one the train-tracks analogy suggests: when the substrate is genuinely deterministic, the system's performance on hard problems is qualitatively different.

A second example, **VIRF** (Verifiable Iterative Refinement Framework, OpenReview October 2025), addresses embodied AI safety. The framework introduces a "tutor-apprentice" architecture in which a deterministic logic tutor, grounded in a formal safety ontology, provides causal feedback to an LLM apprentice. The deterministic tutor is not a guardrail filtering output; it is in the loop, refusing unsafe plans and explaining why in formal terms the LLM can use to repair its plan. The researchers note the standard limitations of pure-LLM approaches: "stochastic nature and lack of formal reasoning capabilities prevent the strict safety guarantees required for physical deployment." The hybrid system supplies the formal guarantees the LLM cannot supply alone.

A third reference point is the family of theorem-proving systems — **Lean**, **Isabelle/HOL**, **HOL Light** — which have been integrated with LLMs to produce formally-verified mathematical proofs. These systems "enforce strict logical correctness via type systems and proof scripts," and a generated proof is either accepted by the type checker or rejected. There is no middle ground, no plausible-looking-but-actually-wrong output. The 2025 ProofNet++ framework illustrates the architecture: an LLM proposes proof tactics, a symbolic verifier checks them, and the verifier's deterministic accept/reject signal drives the search. The output the user receives is not "the LLM is confident this proof is correct" — it is "the proof was machine-checked and is correct, or the system honestly admits it could not produce a proof."

The general pattern across these systems is what the field calls **neurosymbolic transition systems**: every path through the system's state space is determined by allowed symbolic transitions, with non-deterministic choices among the allowed transitions resolved by querying a neural intuition. Inferential moves suggested by the neural component are always checked symbolically for soundness before being accepted. The system as a whole has the operational property that any output it returns has passed a deterministic verification stage and is sound within the expressivity of the available symbolic verifier.

> *This is what a train on tracks looks like in software. The neural component is the engineer suggesting routes; the symbolic component is the track network; no train leaves the track regardless of what the engineer suggests. The trade-off — only the routes that have track laid are reachable — is exactly the trade-off the railway has, and the operational properties are exactly the ones the railway has: scheduled, reliable, no operator improvisation, no possibility of unexpected behavior on the routes that exist.*

The crucial point for buyers and architects is that the neurosymbolic approach is categorically different from the guardrails approach. It is not "more guardrails," not "deeper alignment," not "better fine-tuning." A guardrails system can be made more reliable by adding more layers, but it cannot become a neurosymbolic system by accretion — the architectural commitment is different at the foundation. Either the symbolic component is in the critical path (and constrains what the system can output) or it is downstream (and filters what the system already produced). These are not points on a single spectrum; they are two distinct categories.

The neurosymbolic approach is also categorically different in its limitations. A guardrails system can, in principle, attempt any task its underlying LLM can attempt, with reliability limited by the quality of the guardrails. A neurosymbolic system can only attempt tasks for which a symbolic formalization exists — and the formalization is hard work. This is the train-tracks tradeoff in its most direct form: you can only ride to destinations someone has built track to. For mathematical theorem-proving the tracks exist (decades of formal verification); for embodied robot safety they are being laid (formal safety ontologies); for finance and risk management they exist for some domains (regulatory rules, accounting standards, market microstructure) and do not yet exist for others (qualitative judgment, narrative reasoning). The architecture is only useful where the formal substrate has been built.

## 4. A worked example — a deterministic intelligence application built for risk management

To make the architectural distinction concrete, this section describes a specific deterministic-by-construction application built at Kindynos Advisory for risk-management intelligence work. The example is illustrative of the train-on-tracks architecture and is not a vendor pitch; the purpose is to show what the architecture looks like end to end when it is engineered for genuine determinism rather than retrofitted with guardrails.

### 4.1 The domain and why determinism was the design requirement

The application analyzes how an arbitrary portfolio of financial holdings is exposed to a set of macroeconomic, regulatory, geopolitical, and market-structural events, producing per-holding impact estimates with full provenance: which valuation tier produced each estimate, which probability source produced each event probability,

which causal narrative supports each impact, which source module produced each result, and a determinism anchor (a hash of all inputs) so that any given analysis is reproducible by anyone who re-runs the chain.

The user is a senior risk officer or principal at a financial institution. The downstream consumers of the analysis are investment committees, board risk committees, and regulators. The reliability bar is not "produces a useful-sounding answer most of the time" but "produces the same answer every time, with the same provenance, with no values fabricated by a probabilistic model, with every result traceable back to the inputs that produced it." A hallucinated impact estimate in this context is not a stylistic problem; it is a material misrepresentation to a regulator.

A guardrails architecture is structurally insufficient for this domain. No layer of post-hoc filtering can produce the property the domain requires, which is that a given input always produces a given output by construction. The closest a guardrails system can come is to make hallucinated outputs rare enough that they are unlikely to be encountered by a given user — which is an excellent property for many applications but is not the property the regulator is asking for.

### 4.2 The architecture

The application is built on a deterministic spine. Every output the user sees is produced by classical, rules-based, deterministic software: probability composition follows a published three-tier waterfall (catalog → machine-learning → belief contracts, with composition rules specified in advance); impact estimation follows a tiered valuation model with explicit fallback rules; provenance is captured by reading what each tier produced, not by asking a model to summarize. The system has a **determinism anchor** — a content hash of all inputs — that lets any two runs with the same input be verified identical, and lets any two runs with different inputs be distinguished. The hash is exposed in the user interface so the user can copy it, file it, and reproduce the run later.

Where the system uses language-model output at all, it does so at the edges: summarizing a holding's provenance trace into a paragraph the user can read, or producing a natural-language explanation of a strategy recommendation. The deterministic components produce the operational output; the language model produces the narrative around it. This is the symbolic-core-with-LLM-at-edges pattern the industry literature increasingly recognizes as the mature architecture for high-stakes work.

The provenance model is explicit and exposed in the user interface along four dimensions:

**1. Real versus synthetic-sample data.** Every data point carries a tag stating whether it is real (a real disclosure, a real observed transaction) or a representative sample tagged as such. The tag is visually surfaced as a "sample data" badge. No data is silently presented as real when it is sample.

**2. Valuation and probability tier.** Every probability is labeled with which tier produced it (catalog calibration, ML predictor, belief contract, composed, user

override). Every impact is labeled with which valuation model produced it. When a tier has no authored calibration, the system surfaces the estimate as unavailable or explicitly uncalibrated — tagged with its source — rather than emitting a bare zero that could be misread as a true estimate, or silently falling through to a different tier without saying so.

**3. Source module.** Every impact identifies the backend module that produced it. The user can trace any number on the screen back to the code path that generated it.

**4. Determinism hash.** Every analysis has an inputs_hash that uniquely identifies its inputs; every strategy run has a strategy_run_id; every user session has a workbench_session_id. The chain is exposed in the user interface, monospace, copyable.

A holding-impact in this system is not a number on a screen; it is the number plus the four-dimensional provenance of the number. Whether a user trusts a number depends entirely on what they see in the provenance — and they can see all of it, by design, at every step.

### 4.3 What this architecture buys

The operational properties of the system are the properties the train-tracks analogy predicts. Two runs on the same input produce identical output, byte for byte, hash for hash. There is no “the model gave a different answer this time” — the model is not in the operational path. There is no hallucinated impact estimate, because impact estimates are computed by classical valuation code, not generated by a model. There is no probability silently coerced into the wrong tier, because the tier composition is rules-based and the source tag travels with the number to the screen. There is no “the system is usually right” — the system is either right by construction (because the rules are right) or wrong by construction (because a rule is wrong, in which case the rule can be inspected, debated, and fixed). The failure modes are explicit and bounded.

> *What the architecture does not buy is generality. The system can only analyze portfolios for which the underlying data exists, against ecosystems that have been modeled, using valuation models that have been authored, with probability tiers that have been calibrated. It cannot generalize to a new asset class without that asset class’s valuation model being built. It cannot answer a question about a market it has not been told about. It does not improvise. It is, in the strictest sense, a train: it goes where the tracks have been laid and nowhere else. That is not a limitation to be apologized for. It is the property that gives the system its credibility with the regulator, the investment committee, and the board.*

## 5. The trade-off, honestly stated

Every architectural decision has a trade-off, and the trade-off the train-tracks architecture makes is genuine and worth stating without softening.

A neurosymbolic or deterministic-core system is narrow. It does only what its formal substrate has been built to do. Any task that lies outside the substrate is not "a degraded version of the task the system normally handles" — it is unreachable. A finance application built on a deterministic spine can analyze portfolios of equities, ETFs, and funds because the valuation models for those asset classes have been built; it cannot analyze a portfolio of fine wine without a wine valuation model being authored. A formal-verification system can prove theorems in a domain it has axioms for; it cannot prove theorems in a domain it has not been told about. A regulated-finance application can apply Solvency II rules because Solvency II is a finite, formal regulatory regime; it cannot apply "general financial prudence" because that is not a formal regime.

This narrowness has three honest consequences. First, **building the substrate is expensive**. Authoring valuation models, calibrating probability tiers, writing formal safety ontologies, encoding regulatory rules into machine-checkable form — each of these is real engineering work that an LLM-with-guardrails system mostly avoids. The cost of a deterministic system is front-loaded into the substrate. Second, **adding capability is structural, not incremental**. A new asset class is not a parameter change; it is a new model and a new set of tracks. The system grows by deliberate extension, not by emergent learning. Third, **the system cannot handle truly open-ended questions** — the kind that emerge constantly in qualitative analysis, that mix domains, that involve judgment about whether a given framing is even the right one. For those questions the train-tracks architecture is the wrong tool; an LLM with good context and a thoughtful user is the right tool.

The literature is honest about the converse trade-off too. A respected analysis of AI agent brittleness notes: "Agents fail at basic UI navigation, struggle with pop-ups, and exhibit cascading failures where errors in one component bring down entire systems. Unlike traditional software with predictable failure modes, agent unpredictability makes them unsuitable for mission-critical applications."

Stated baldly: the highway architecture has the property that a vehicle can go almost anywhere, with the cost that it can also go somewhere wrong. The train architecture has the property that the vehicle can only go where the tracks are laid, with the benefit that it cannot go somewhere wrong. Neither architecture is "better." They are different tools for different problems. The error the industry makes is suggesting that one architecture, with enough engineering effort, becomes the other. It does not. A highway with enough guardrails is a very safe highway; it is not a railway.

## 6. When each architecture is correct

The natural follow-on question is: for any given problem, how does an architect decide which architecture to use? The answer is not abstract; it reduces to a small set of design questions that, when answered honestly, point to one of the two architectures.

**Is the cost of a single deviation catastrophic or tolerable?** If a wrong output causes material harm — a misstated risk to a regulator, an unsafe instruction to a

robot, an incorrect medication dosage — the architecture must be the kind that cannot fabricate outputs outside its encoded rules — so that any error is located in an inspectable rule, model, or input rather than hidden inside a stochastic generation process. If a wrong output is correctable, cosmetic, or one of many produced in volume — a slightly off summary, an imperfect translation, a sub-optimal email draft — the guardrails architecture is appropriate and far cheaper.

**Is the domain formally specifiable, or essentially open-ended?** If the rules of the domain can be written down — regulatory regimes, accounting standards, market microstructure, formal proof systems, safety ontologies — the deterministic architecture is buildable and the train tracks can be laid. If the domain is open-ended, requires aesthetic or qualitative judgment, or mixes domains in unpredictable ways, the formalization is not feasible and the LLM is the right architecture.

**Does the user need reproducibility?** If a result must be reproducible by a regulator, an auditor, a counterparty, or a future version of the user themselves, the architecture must produce identical output for identical input, every time. This is the property a determinism anchor (a content hash of inputs) makes verifiable. A guardrails architecture cannot offer this in the strict sense — temperature, sampling, and the inherent statistical nature of the model mean that "the same prompt" almost never produces "the same answer" character-for-character.

**Is the surface area bounded?** A deterministic system can only cover the surface area it has been built for. If the user genuinely needs to handle inputs the designers cannot anticipate, an LLM is the appropriate tool. If the inputs are bounded by the domain (a portfolio is a portfolio, a balance sheet is a balance sheet, a theorem statement is a theorem statement), the deterministic architecture's narrowness is not a constraint.

**What is the failure mode?** A guardrails system fails gracefully but unpredictably — it usually produces something useful, sometimes produces something wrong, and the engineering effort is directed at the rate of the latter. A deterministic system fails bluntly but predictably — it either produces the right answer or honestly admits it cannot, with the boundary visible to the user. For regulated work, predictable failure is usually preferable to graceful failure.

> *A clear way to state the rule: the deterministic-core architecture is the correct choice when the cost of a single false positive (a wrong output presented as correct) exceeds the cost of a high false negative rate (the system honestly admitting it cannot handle a case). The guardrails architecture is the correct choice when the converse is true. The industry's current default — wrap an LLM in checks and call it deterministic — solves the second problem and pretends to solve the first.*

## 7. Implications for the AI industry

If the architectural distinction this paper draws is correct, four implications follow that the industry has so far been slow to acknowledge.

**First, the major AI platforms have selected an architecture that is not deterministic in the train-tracks sense, and they will not become so by incremental improvement.** The cloud-AI platforms — including the offerings from the largest providers — are built around a probabilistic core (the LLM) with deterministic engineering around it (function calling, structured output, retrieval, evaluation harnesses, content filters, alignment training). The architecture is sophisticated and constantly improving, and it is right for an enormous range of problems. But it is not the architecture that produces the train-tracks property, and no amount of further investment in better guardrails will convert it into that architecture. Buyers who need genuine determinism will not get it by buying a more advanced version of this stack.

**Second, the neurosymbolic and hybrid approaches are not "future research" but increasingly mature production architectures, and they remain underrepresented in the major platform marketing.** The literature surveyed in §3 is not speculative; it describes systems being deployed in finance, healthcare, regulatory compliance, and embodied robotics today. The Kognitos brain architecture, the Lean-integrated theorem provers, the formal-verification stacks for safety-critical systems, the rules-based commission engines with LLM-augmented insights — these are operational. The market has been slow to recognize them as architecturally distinct from "AI with guardrails," partly because the marketing vocabulary is not yet sharp enough to distinguish them. This is changing, and the distinction is likely to become more visible as enterprises in regulated industries discover that their reliability requirements cannot be met by the highway-with-guardrails option.

**Third, "deterministic guardrails" as currently marketed is a useful technology but should not be confused with deterministic systems.** Buyers reading vendor materials in 2026 should ask a sharp diagnostic question: is the deterministic component in the *critical path* (the system's output is produced by the deterministic component, with the LLM as a suggestion engine) or is it in the *downstream filter* (the system's output is produced by the LLM, with the deterministic component checking it for compliance)? The answer determines whether what is being sold is the train-tracks architecture or the highway-with-guardrails architecture. The marketing language tends to be the same; the architectures are different.

**Fourth, the bifurcation in the market is likely to deepen rather than resolve.** The guardrails architecture continues to serve a large and growing range of applications where the trade-offs favor flexibility over absolute determinism — content generation, summarization, conversational interfaces, code assistance, qualitative analysis. The deterministic-core architecture continues to be built out for applications where the trade-offs are reversed — regulated risk management, safety-

critical robotics, formal verification, compliance automation, financial reporting. The future is not “one wins”; it is two architectures coexisting, with the appropriate one selected for the problem. The work of the next several years is for buyers, architects, and the technical press to develop the vocabulary that distinguishes them precisely.

## 8. Conclusion — saying what we built

The motivation for this paper was concrete. At a recent industry conference in Bangkok, I watched a procession of platform vendors describe reliability, safety, alignment, and “deterministic infrastructure” — and recognized that the architecture being marketed under those terms was not the architecture we had built. The vendors’ systems were highways with very good guardrails. The Kindynos system was a train on tracks. The vocabulary of the conference did not contain a clean way to draw that distinction; the purpose of this paper is to draw it.

The architectural option of building a system that, by construction, can only produce specific outputs in specific ways for specific inputs — that has formal provenance for every value displayed, a determinism anchor that lets every analysis be reproduced, and a sharply bounded surface area outside of which it honestly admits it cannot help — is a real architectural choice, present in the academic literature under the names neurosymbolic AI and hybrid AI, present in production in regulated industries, and architecturally distinct from anything the major AI platforms currently sell as “deterministic.” It is harder to build than wrapping an LLM in guardrails. It is narrower in what it can do. It is the right architecture for problems where the cost of unpredictability is unacceptable.

> *To use the analogy one final time: the major platforms are building progressively better highways. Some of those highways are extraordinary engineering achievements. They are still highways. A risk-management report for a regulator should not ride on a highway, however good. It should ride on a train, on tracks the operator laid, to a destination the operator knows. That is what genuine determinism looks like, and that is what an architecture for high-stakes work must deliver. The work of the field, for the operators who need the second architecture, is to lay more track.*

## Sources & References


[1] Rahaman, Z. (March 2026). LLMs Should Reason. Infrastructure Should Enforce. Data Science Collective / Medium. https://medium.com/data-science-collective/llms-should-reason-infrastructure-should-enforce-86493b936f84

[2] Tools for Building Deterministic LLM Systems. DZone (February 2026). https://dzone.com/articles/tools-for-building-deterministic-llm-systems

[3] Venkata, A. (April 2026). LLM Guardrails and Safety in Production AI Systems. Medium. https://medium.com/@advenkata/llm-guardrails-and-safety-in-production-ai-systems-1375d44be3ef

[4] Verma, A. (December 2025). The Ultimate Guide to Guardrails in GenAI. Medium. https://medium.com/@ajayverma23/the-ultimate-guide-to-guardrails-in-genai-securing-and-standardizing-llm-applications-1502c90fdc72

[5] Nunepalli, B. (April 2026). From Harness to Enforcement: Designing Deterministic Guardrails for LLM Systems. Medium. https://bh3r1th.medium.com/from-harness-to-enforcement-designing-deterministic-guardrails-for-llm-systems-6a9912ba7eba

[6] Avinash, K., Pareek, N., Hada, R. (FutureAGI). Protect: Towards Robust Guardrailing Stack for Trustworthy Enterprise LLM Systems. arXiv 2510.13351. https://arxiv.org/pdf/2510.13351

[7] Engineering Determinism: Practical Strategies for Reliable LLM Applications. Zartis (January 2026). https://www.zartis.com/engineering-determinism-practical-strategies-for-reliable-llm-applications/

[8] What is Neurosymbolic AI? The Technology Behind Hallucination-Free Automation. Kognitos (March 2026). https://www.kognitos.com/blog/what-is-neurosymbolic-ai/

[9] Nezhad, S. B., Li, Y., & Agrawal, A. (2026). SymCode: A Neurosymbolic Approach to Mathematical Reasoning via Verifiable Code Generation. arXiv:2510.25975v2; EACL 2026 Findings. https://arxiv.org/abs/2510.25975

[10] ProofNet++: A Neuro-Symbolic System for Formal Proof. arXiv 2505.24230 (May 2025). https://arxiv.org/pdf/2505.24230

[11] Wu, F., Zheng, X., Qu, Y., Wang, Z., Feng, Z., & Li, H. Grounding Generative Planners in Verifiable Logic: A Hybrid Architecture for Trustworthy Embodied AI (VIRF). OpenReview wb05ver1k8 (October 2025); arXiv:2602.08373. https://openreview.net/forum?id=wb05ver1k8

[12] A Comparative Study of Neurosymbolic AI Approaches to Interpretable Logical Reasoning. arXiv 2508.03366 (August 2025). https://arxiv.org/html/2508.03366

[13] Avoiding LLM Hallucinations: Neuro-symbolic AI and other Hybrid AI approaches. Cota Capital (March 2026). https://www.cotacapital.com/knowledge-base/avoiding-llm-hallucinations-neuro-symbolic-ai-and-other-hybrid-ai-approaches/

[14] Domain-Specific Models: 7 Powerful AI Advantages. Progressive Robot (April 2026). https://www.progressiverobot.com/2026/04/28/domain-specific-models/

[15] When Smaller, Domain-Specific AI Models Beat Giant LLMs. Appsvolt (February 2026). https://appsvolt.com/when-smaller-domain-specific-ai-models-beat-giant-llms-a-decision-framework-for-product-companies/

[16] Domain-Specific LLMs vs. General AI: Why Niche Models Win. Qodequay (February 2026). https://www.qodequay.com/domain-specific-llms-enterprise

[17] The fundamental limitations of AI agent frameworks. Medium (July 2025). https://medium.com/@thekrisledel/the-fundamental-limitations-of-ai-agent-frameworks-expose-a-stark-reality-gap-7571affb56e5

[18] Revising human-systems engineering principles for embedded AI applications. NIH / PMC. https://pmc.ncbi.nlm.nih.gov/articles/PMC10790827/

[19] Estimating the Brittleness of AI: Safety Integrity Levels and the Need for Testing Out-Of-Distribution Performance. arXiv 2009.00802. https://arxiv.org/pdf/2009.00802

---

**Note on sources and currency.** *This whitepaper draws on a mix of academic literature (arXiv preprints, OpenReview submissions, peer-reviewed AI-safety research), industrial-technical analysis (production-engineering blogs, vendor architectural documentation), and current AI-platform marketing materials examined critically. Citations are current as of preparation in May 2026 and reflect an actively moving field; vocabulary continues to evolve and some specific positions described here may shift. The paper is research and strategic commentary, not legal, tax, investment, accounting, or regulatory advice.*