The Spine: A Construction-Not-Trace Architecture for Auditable Machine Reasoning

The Compiler for Machine Reasoning

Structural Verification as an Alternative to Inspection-Based AI Governance

---

Authors: Stephen Fishburn; Larry Signorile **Affiliation:** Kenshiki Labs **Correspondence:** stephen@kenshiki.ai; larry@kenshikilabs.com **Version:** Working paper, draft of May 1, 2026 **Status:** Pre-print. Comments welcome.

Abstract

Contemporary AI governance is predicated on inspection — grading the probabilistic outputs of models that were never engineered to produce structural quality at production time. This paper argues the paradigm is a category error: quality cannot be inspected into a probabilistic guess any more than it could be inspected into a defective automobile in 1955 Detroit. We present a deterministic architecture for machine reasoning that builds quality in at production time, treating the frontier model as an untrusted candidate generator and a downstream kernel as the deterministic compiler that admits inferences only when they can be structured as typed proof graphs anchored to admitted facts. The architecture composes four intellectual lineages with pre-deep-learning foundations and live modern descendants — a *semantic engine* (referential identity, model-relative satisfaction, prose-to-structure mapping), a *deterministic core* (typed kernel with refusal as a first-class output), a *structural audit* (typed defeasible inference legible to non-expert reviewers), and a *differentiated-assurance* layer (environment-specific failure-mode contracts) — and ships them as a single deployable runtime. The contribution is architectural rather than algorithmic: under stated admission rules, hidden-premise hallucinations are *un-compilable* rather than merely catchable after the fact. Inspection regimes scale linearly with deployment volume; construction-time admission pays a fixed cost and emits reusable artifacts. The word *compiler* is used in a specific technical sense defined in §1.2.

Keywords: AI governance; deterministic reasoning; defeasible inference; argumentation schemes; assurance cases; regulatory technology; structural verification; hallucination; auditability; large language models.

ACM CCS: Computing methodologies → Knowledge representation and reasoning → Reasoning about belief and knowledge; Computing methodologies → Artificial intelligence → Philosophical/theoretical foundations of artificial intelligence; Software and its engineering → Software verification and validation.

JEL Classification: K23 (Regulated Industries and Administrative Law); O33 (Technological Change: Choices and Consequences); D81 (Criteria for Decision-Making under Risk and Uncertainty).

1. Introduction

The AI safety industry is doing rigorous work inside the wrong paradigm. Every major governance approach today — from RLHF at training time, to red-teaming and evaluations at evaluation time, to conformity assessments and audit trails at deployment — is inspection. The work grades the probabilistic output of a process that was never engineered to produce structural quality in the first place. This is not a flawed approach to safety. It is a category error about where quality comes from. The alignment researchers and policy officers running these programs are not the failure; the paradigm is. You cannot inspect quality into a probabilistic guess.

The historical precedent is structurally instructive. Detroit ran assembly lines that produced defects by design and spent billions on quality control at the end of the line to catch them before shipment. Toyota engineered the line so the defects could not occur, importing W. Edwards Deming's mandate to *cease dependence on inspection* and build quality into the act of production [1]. Detroit lost industries it had invented. The collapse took thirty years and was visible in advance to anyone reading Deming. The AI industry is currently recreating 1955 Detroit at a faster clock speed and with higher liability exposure.

This paper describes an alternative architecture. The frontier model is decoupled from the final output and treated as an untrusted candidate generator: it proposes inferences, declares the load-bearing bridges between evidence and conclusion, and surrenders its proposals to a deterministic kernel. The kernel admits typed facts into a closed register, requires every load-bearing bridge to anchor to admitted facts, discharges regulatory burdens against typed criteria, and refuses to compile any conclusion that depends on a hidden premise. The output is not an answer that has been graded. It is an artifact that compiled, or it does not exist.

The architecture is not a better safety filter. It is a compiler (in the technical sense made precise in §1.2) that makes hidden-premise hallucinations *un-compilable under the kernel's stated admission rules*, not merely catchable downstream. Inspection regimes scale linearly with deployment volume, and at frontier-model volumes the per-decision inspection cost approaches the per-decision value of the inspected output. When the inspection-cost-to-output-value ratio crosses unity at scale, architectures that pay a fixed admission cost at construction time and emit reusable compiled artifacts will be advantaged over architectures that pay an inspection cost on every decision and re-incur it whenever the threat model or the judge is updated. The strongest live counter-argument to that scaling claim — that LLM-as-judge automated evaluation drives the marginal cost of inspection toward zero — is addressed directly in §1.1 and §6.1.

The paper proceeds in four sections, each devoted to one of the intellectual lineages the architecture composes. Section 2 (the semantic engine) describes what the compiler operates on; Section 3 (the deterministic core) describes the kernel that does the operating; Section 4 (the structural audit) describes how the kernel's output survives review; Section 5 (differentiated assurance) describes how the same compiler ships into environments with incompatible failure tolerances. Section 6 integrates the lineages and presents the system architecture diagram. Section 7 addresses falsifiability; §7.1 enumerates limitations and threats to validity. Section 8 concludes. A short *Related Work* section follows this introduction.

Contributions. The specific contributions of this paper are: (i) a definition of *compiler* for machine reasoning as a deterministic transform $K : \mathcal{C} \to \mathcal{A} \cup \{\bot\}$ with admission soundness, refusal totality, and structurally locatable refusal causes (§1.2); (ii) a four-lineage decomposition that names what each lineage supplies as a runtime requirement rather than as historical homage (§§2–5); (iii) the *bracketed-LLM* architectural pattern — frontier model upstream-bounded by catalog-driven retrieval and downstream-bounded

[1] Deming, W. E. (1982). *Out of the Crisis*. MIT Center for Advanced Engineering Study.

by typed admission — as the integration point (§6); (iv) the *differentiated-assurance* deployment-tier model (Studio fails soft, Refinery fails closed, Clean Room fails hard) as the correct architectural granularity for assurance-case authoring in regulated AI (§5); (v) a structural argument, in §6.1, for why LLM-as-judge automated evaluation does not close the inspection gap, with a cost-crossover sketch in §6.1.1; and (vi) an enumerated falsifiability surface (§7) with explicit limitations and threats to validity (§7.1). A minimal worked trace illustrating the kernel's four outcomes is given inline in §2.1 and expanded in Appendix A; a concrete regulatory worked example (ECOA adverse-action notice) appears in Appendix A.7.

1.1 Related Work

This paper sits at the intersection of four conversations. We name each, identify the closest contributions, and locate the present work relative to them.

Inspection-paradigm AI safety. The paradigm critiqued here is articulated most fully in the contemporary AI-safety literature on evaluation [2], red-teaming [3], and constitutional alignment [4]. Each accepts the probabilistic-model-with-inspection-overlay architecture as given and proposes refinements to the inspection layer. The present work rejects that framing in the architectural sense Deming rejected end-of-line quality control: the inspection programs are rigorous, but they are inspecting an artifact whose quality is not structurally constructible at production time. Closest in spirit on the pre-deep-learning side are the symbolic-AI revival arguments of Garcez and Lamb on neurosymbolic systems [5] and Marcus and Davis on the limits of pure deep learning [6]; both argue that statistical learning alone is insufficient for systems that must reason. Neither describes a deployable compiler architecture; this paper does.

Runtime governance for AI agents. A distinct and rapidly growing literature addresses governance *at runtime* for agentic AI systems — observing agent behavior in deployment, detecting drift from declared policy, enforcing policy at the model-call boundary, and producing conformance evidence. MI9 [7] is the most fully described runtime governance framework in this corpus, instrumenting agent execution to detect divergence from declared intent. Aegis [8] formalizes cryptographic runtime policy enforcement for autonomous agents under regulatory constraint. AGENTSAFE [9] specifies a unified framework for ethical assurance and governance of agentic AI. Policy Cards [10] and Policy-as-Prompt [11] both translate human-readable policy into machine-enforceable artifacts that condition agent behavior. Kaptein, Khan, and Podstavnychy's *Runtime Governance for AI Agents: Policies on Paths* [12] consolidates the runtime-governance pattern as a discipline

[2] Bommasani, R., Hudson, D. A., Adeli, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[3] Perez, E., Huang, S., Song, F., et al. (2022). Red teaming language models with language models. *Proceedings of EMNLP 2022*, 3419–3448.

[4] Bai, Y., Kadavath, S., Kundu, S., et al. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

[5] Garcez, A. d'A., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(11), 12387–12406.

[6] Marcus, G., & Davis, E. (2019). *Rebooting AI: Building Artificial Intelligence We Can Trust*. Pantheon.

[7] Wang, C. L., Singhal, T., Kelkar, A., & Tuo, J. (2025). MI9: An integrated runtime governance framework for agentic AI. *arXiv preprint arXiv:2508.03858*.

[8] Mazzocchetti, A. M. (2026). Cryptographic runtime governance for autonomous AI systems: The Aegis architecture for verifiable policy enforcement. *arXiv preprint arXiv:2603.16938*.

[9] Khan, R., Joyce, D., & Habiba, M. (2025). AGENTSAFE: A unified framework for ethical assurance and governance in agentic AI. *arXiv preprint arXiv:2512.03180*.

[10] Mavračić, J. (2025). Policy Cards: Machine-readable runtime governance for autonomous AI agents. *arXiv preprint arXiv:2510.24383*.

[11] Kholkar, G., & Ahuja, R. (2025). Policy-as-Prompt: Turning AI governance rules into guardrails for AI agents. In *Proceedings of the 3rd Regulatable ML Workshop, NeurIPS 2025*. *arXiv preprint arXiv:2509.23994*.

[12] Kaptein, M., Khan, V.-J., & Podstavnychy, A. (2026). Runtime governance for AI agents: Policies on paths. *arXiv preprint arXiv:2603.16586*.

and formalizes path-level policy enforcement on agent trajectories. Koch's *From Governance Norms to Enforceable Controls* [13] addresses the layered translation problem from regulation to runtime guardrails. Kurshan and colleagues' *Agentic Regulator* [14] frames the regulator's role in agentic-AI ecosystems for finance. Marin and Chaudhary [15] argue for adaptive runtime governance under partial observability of autonomous agents.

The Compiler thesis is structurally distinct from this corpus and composes with it rather than competing. The runtime-governance literature operates on the *agent-behavior surface* — observing what an agent did, authorizing what it may do next, and enforcing policy at the model-call or tool-call boundary against the trace it produces. The architecture here operates one layer earlier, on the *inference-construction substrate*: the kernel refuses to compile inferences that depend on hidden premises, so the agent is constrained in what it can do before it does anything observable. The two layers compose. A runtime governance system can supervise an agent whose underlying inferences are themselves typed proof graphs anchored to admitted facts; the runtime layer then governs *behavior over time* while the construction layer governs *inferences as they are produced*. Conversely, the construction layer alone does not govern the agent's loop, its tool use, or its long-horizon planning — runtime governance is needed for that. We argue that production-grade agentic AI in regulated domains will require both layers, and that the absence of either is the gap the present literature has not yet closed. The Compiler paper contributes the construction layer.

Defeasible-reasoning implementations. The argumentation-theory tradition has produced computational implementations — CARNEADES [16] and ASPIC+ [17] — that operationalize defeasible reasoning at a research-prototype scale. Neither is engineered for regulated deployment, neither integrates with frontier-LLM proposers, and neither addresses the assurance-case discipline that distinguishes deployment environments. The kernel described here is informed by their formal models but is not derived from their codebases.

Assurance cases applied to AI. The assurance-case literature [18] [19] [20] is mature in aviation, nuclear, and medical-device certification, and has begun to be applied to AI systems specifically [21] [22]. The differentiated-tier model in Section 5 is a direct application of GSN to the AI-deployment problem; the contribution is not the GSN methodology itself but the discipline of shipping three structurally distinct assurance contracts (Studio, Refinery, Clean Room) as architectural features rather than as deployment configurations.

Output guardrails and constrained decoding. A separate engineering tradition addresses the same goal — preventing bad outputs from reaching production — at a different layer of the stack. Shielded reinforcement

[13] Koch, C. (2026). From governance norms to enforceable controls: A layered translation method for runtime guardrails in agentic AI. *arXiv preprint arXiv:2604.05229*.

[14] Kurshan, E., Byrd, D., et al. (2025). The agentic regulator: Risks for AI in finance and a proposed agent-based framework for governance. *arXiv preprint arXiv:2512.11933*.

[15] Marin, G., & Chaudhary, J. (2026). Governing what you cannot observe: Adaptive runtime governance for autonomous AI agents. *arXiv preprint arXiv:2604.24686*.

[16] Gordon, T. F., Prakken, H., & Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15), 875–896.

[17] Modgil, S., & Prakken, H. (2014). The ASPIC+ framework for structured argumentation: A tutorial. *Argument and Computation*, 5(1), 31–62.

[18] Kelly, T. P. (1998). *Arguing safety: A systematic approach to managing safety cases* (Doctoral dissertation). University of York, Department of Computer Science.

[19] Kelly, T., & Weaver, R. (2004). The Goal Structuring Notation — A safety argument notation. In *Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases*.

[20] Assurance Case Working Group. (2021). *Goal Structuring Notation Community Standard, Version 3*. Safety-Critical Systems Club.

[21] Bloomfield, R., & Rushby, J. (2021). Assurance 2.0: A manifesto. In *Proceedings of the 29th Safety-Critical Systems Symposium*.

[22] Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., & Habli, I. (2021). Guidance on the assurance of machine learning in autonomous systems (AMLAS). *University of York Assuring Autonomy International Programme*.

learning [23] composes a learned policy with a runtime safety shield that vetoes unsafe actions before they execute; constrained decoding frameworks (LMQL [24], Outlines [25], guidance/JSON-mode constraints) restrict the LLM's token-level output space to grammars or schemas declared in advance; production guardrail systems (NeMo Guardrails [26], Llama Guard [27], the OWASP LLM Top 10 mitigations) bracket the model with input/output classifiers and policy checks; general-purpose policy engines (OPA [28] and Cedar [29]) provide policy-as-code substrates that several governance products compose with model serving. These systems are real, deployed at scale, and represent the strongest live alternative to the construction-layer architecture proposed here. We acknowledge them and distinguish on a single axis: shields, decoders, classifiers, and policy engines all operate on *the model's surface output* (tokens, actions, structured strings) at runtime, after the inference is generated. The construction layer operates on *the inference itself* before the output is composed: the kernel refuses to compile candidate inferences that depend on hidden premises, regardless of whether the resulting tokens would have passed a shield, satisfied a grammar, or matched a policy. The two approaches compose — a deployed system can use both — but they are not substitutes. Output guardrails answer *did the model produce a permissible string*; the construction layer answers *is the inference structurally anchored to admitted facts*. The first is a runtime filter; the second is a production-time discipline. Inspection-collapse arguments (§1) and judge-substitution arguments (§6.1) apply to all surface-level filtering with the same force; they do not apply to typed admission.

Retrieval. The catalog-driven retrieval discipline introduced briefly in Section 6 inverts the standard retrieval-augmented generation pattern [30]; a fuller treatment is out of scope for this paper.

Automated evaluation and LLM-as-judge. A parallel research program treats LLMs themselves as evaluators of model output [31] [32] [33]. The premise is that automated scoring drives the marginal cost of inspection toward zero, defeating the inspection-collapse argument made in §1. We engage this counter-argument at length in §6.1; in brief, automated judges reduce the human cost of grading but do not manufacture the structural guarantees the construction layer supplies, and they are vulnerable to documented failure modes — positional bias, self-preference, length bias, sycophantic agreement, and distributional drift between judge and judged — that compound with deployment volume rather than amortizing over it. Judges and constructors are complements, not substitutes.

[23] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., & Topcu, U. (2018). Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

[24] Beurer-Kellner, L., Fischer, M., & Vechev, M. (2023). Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI), 1946–1969.

[25] Willard, B. T., & Louf, R. (2023). Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

[26] Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., & Cohen, J. (2023). NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. *arXiv preprint arXiv:2310.10501*.

[27] Inan, H., Upasani, K., Chi, J., et al. (2023). Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*.

[28] Sandall, T. (2019). Open Policy Agent: Policy-based control for cloud native environments. Cloud Native Computing Foundation. https://www.openpolicyagent.org/

[29] Bornholdt, J., Brandt, C. E., Cook, B., et al. (2024). Cedar: A new language for expressive, fast, safe, and analyzable authorization. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA1).

[30] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

[31] Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36.

[32] Gu, J., Jiang, X., Shi, Z., et al. (2024). A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*.

[33] Li, H., Dong, Q., Chen, J., et al. (2024). LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

1.2 What "Compiler" Means in This Paper

The word *compiler* is load-bearing in this paper and is used in a specific technical sense that differs from the Aho-Sethi-Ullman sense familiar to programming-language readers [34]. We anchor the term here so that subsequent claims — especially the un-compilability claim — have a definite referent.

A compiler in this paper is a deterministic transform $K : \mathcal{C} \to \mathcal{A} \cup \{\bot\}$ from a space of *typed candidate inferences* $\mathcal{C}$ — produced by the LLM proposer over a closed Fact Register F and a typed Catalog Registry R — to either a *compiled artifact* $a \in \mathcal{A}$ (a signed, typed directed acyclic graph whose leaves are facts in F, whose edges are bridges drawn from R, and whose root is the conclusion) or a refusal $\bot$. The transform is total: every input $c \in \mathcal{C}$ either compiles to a compiled artifact a — *canonicalized* under a declared canonicalization procedure (lexicographic node ordering, scheme-instantiation normalization, and provenance-edge deduplication, with the field-level serialization step performed under the JSON Canonicalization Scheme of RFC 8785 [35]) so that the artifact is unique up to that canonicalization — or refuses with a structurally locatable reason ($\bot_{\text{missing-fact}}$, $\bot_{\text{unmet-critical-question}}$, $\bot_{\text{cycle}}$, $\bot_{\text{type-mismatch}}$, $\bot_{\text{unanchored-bridge}}$).

Two conditions specify the kernel's intended correctness. **Admission soundness:** if $K(c) = a$, then every leaf of a is a fact admitted into F under the *canonical reference identity* relation that the Fact Register enforces (a Fregean sense/reference discipline applied at admission time — see §2), every edge of a instantiates a Walton-style argumentation scheme [36] [37] drawn from R and evaluated under Dung–Prakken attack/defeat semantics [38] [39] whose required critical questions are *discharged* (either answered against admitted facts or formally accepted by a declared burden-of-proof allocation in the Carneades sense [40]), and the support graph from leaves to root is acyclic. Compiled artifacts emitted as regulated outputs (Refinery and Clean Room tiers, §5) require all required critical questions to be answered against admitted facts; intermediate graphs may carry critical questions that are explicitly marked open with a declared burden allocation, but those graphs do not cross the regulated-output boundary. **Refusal totality (no-silent-failure):** if any of these conditions fails for c, then $K(c) = \bot$ with the corresponding structurally locatable reason; no candidate ever passes silently. The term *refusal totality* is preferred to the more common logical sense of *completeness* (every truth is provable) to avoid collision with the standard term of art. The full proof — including the formal model of admission, the bridge-typing judgment, the canonicalization procedure, and the assumptions under which soundness holds — is out of scope for this paper, which establishes the architectural specification rather than its mechanized verification. The present paper relies on the existence of K as a specification, not on its mechanized proof.

Three implications matter for the rest of the argument:

1. *"Mathematically un-compilable"*, where it appears below, means *un-compilable under the kernel's stated admission rules*, given a correct Fact Register and a correct Catalog Registry. It is a statement about K, not about a Turing-complete language or about computability in the Church-Turing sense.

[34] Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Addison-Wesley.

[35] Rundgren, A., Jordan, B., & Erdtman, S. (2020). *JSON Canonicalization Scheme (JCS)*. RFC 8785, Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc8785

[36] Walton, D. (1996). *Argumentation Schemes for Presumptive Reasoning*. Erlbaum.

[37] Walton, D., Reed, C., & Macagno, F. (2008). *Argumentation Schemes*. Cambridge University Press.

[38] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2), 321–357.

[39] Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2), 93–124.

[40] Gordon, T. F., Prakken, H., & Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15), 875–896.

The strength of the claim is bounded by the correctness of F and R, which is the falsifiability surface §7 enumerates.

2. The compiler is *not* a theorem prover, *not* a constraint solver, and *not* a Datalog engine, though it borrows judgment-checking machinery from each. It is closest in spirit to a typed admission gate with a structured-argument output, in the lineage of CARNEADES [41] and ASPIC+ [42] but engineered for production deployment.
3. The choice of *compiler* over *verifier*, *gate*, or *policy engine* is deliberate: the artifact a is a *constructed object* with declared structure, not a *passed* or *failed* evaluation of a pre-existing object. The construction discipline is what distinguishes the architecture from runtime guardrails (§1.1) and from LLM-as-judge automated evaluation (§6.1).

2. The Semantic Engine

A compiler that operates on meaning needs meaning to be a thing it can operate on. The probabilistic stack treats meaning as a distribution over tokens — a soft cloud whose center happens to look right when the model is trained well and looks wrong when it is not. A cloud is not compilable. A compiler requires a unit of meaning that has identity, an entailment relation it can check, and a syntax-to-semantics mapping that executes. Each is a partially solved problem with a mature literature behind it; each was founded before the deep learning era and survives today in working formal-semantics and semantic-parsing traditions.

Frege solved identity. The 1892 distinction between *sense* and *reference* [43] is the discipline that forces *the morning star* and *the evening star* to resolve to a single referent in the register, not to two probabilistically adjacent strings. The framework's atomic de-referencing rule is Frege made operational: a fact has one identity in the register; linguistic variation collapses to that identity at admission, or the fact does not enter. There is no fuzzy-match tolerance, no embedding-distance threshold, no "close enough." The semantic distance between *the morning star* and *the evening star* is zero or undefined; it is never small. The Fregean discipline survives in modern formal semantics [44] and in description-logic-based knowledge representation [45], but it is precisely the discipline that vector-similarity retrieval discards.

Tarski solved truth. The 1933 model-relative satisfaction relation [46] [47] gives the compiler something it can actually check: a statement is *satisfied in a model* if it bears the right structural relation to the facts admitted into that model. This is the move that lets the framework ship into environments where ground truth is contested — legal classification, medical diagnosis, intelligence analysis, regulatory enforcement. The compiler is not making a correspondence claim about the world. It is making a structural-admissibility claim against a defined Fact Register: the candidate inference is admissible if its support graph is derivable, edge by typed edge, from facts admitted into F under the bridge inventory R. The kernel checks structural

[41] Gordon, T. F., Prakken, H., & Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15), 875–896.

[42] Modgil, S., & Prakken, H. (2014). The ASPIC+ framework for structured argumentation: A tutorial. *Argument and Computation*, 5(1), 31–62.

[43] Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100, 25–50. English translation: On sense and reference, in P. Geach & M. Black (Eds.), *Translations from the Philosophical Writings of Gottlob Frege* (1952). Blackwell.

[44] Heim, I., & Kratzer, A. (1998). *Semantics in Generative Grammar*. Blackwell.

[45] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[46] Tarski, A. (1933). Pojęcie prawdy w językach nauk dedukcyjnych [The concept of truth in the languages of the deductive sciences]. *Prace Towarzystwa Naukowego Warszawskiego*, Wydział III, 34. English translation in *Logic, Semantics, Metamathematics* (1956). Oxford University Press.

[47] Tarski, A. (1944). The semantic conception of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, 4(3), 341–376.

admissibility and not model-theoretic entailment in the full first-order sense; the relevant judgment is closer to *derivability under the bridge inventory* than to $\Gamma \models \phi$ over an unrestricted model class. If the auditor disputes a conclusion, the dispute is structurally locatable: either an admitted fact is wrong, or a bridge type was misapplied. The framework cannot fail silently in the way *the model hallucinated* fails; any failure is localized either to a misadmitted fact in F or to a mis-typed bridge in R, and both are objects an auditor can name and find.

Montague closed the loop. The 1970 demonstration [48] [49] that natural language has executable mathematical semantics is the existence proof that what the LLM produces in prose can be compiled at all. Without Montague, the framework would be making a metaphysical claim — *language has structure* — and asking the reader to take it on faith. With Montague, the claim is technical: the structure is recoverable, the mapping from syntax to semantics is computable, and the prose the LLM emits can be reduced to a typed graph without loss of the load-bearing content. Montague did not have the extraction technology to make this practical at scale. The deep learning era produced the extraction technology and ignored the compositional target. The framework reunites them.

Frege/Tarski/Montague are the foundational figures; the working modern lineage descends from them. Kamp's Discourse Representation Theory [50] extended Tarski-Montague compositional truth to multi-sentence discourse with anaphora and donkey sentences that the original framework could not handle; dynamic semantics [51] formalized how meaning updates across an unfolding text rather than being assigned statically per sentence; the contemporary semantic-parsing tradition [52] [53] produced executable mappings from natural-language input to logical-form output that are the working ancestors of the framework's prose-to-DAG extractor. The mature versions are richer than the foundational originals in every dimension a working semanticist would test: they handle context-dependence, presupposition projection, generalized quantifiers, and graded entailment that Frege-Tarski-Montague cannot natively express. The framework draws on the modern formalisms for its compositional machinery and its handling of cross-sentential reference; what it imports from Frege-Tarski-Montague specifically is the foundational discipline that compositional truth is a *thing one can compute*, against which the probabilistic stack's treatment of meaning as a distribution over tokens is a regression rather than a refinement.

What the compiler requires from this lineage is the existence of a coherent object to compile. Frege gives it canonical reference identity. Tarski gives it a model-relative satisfaction relation against which structural admissibility can be checked. Montague gives it the bridge from prose to structure. The modern descendants give it the discourse-level and lexical-coverage machinery the foundational moves did not. Without all of these, the kernel has nothing to operate on.

2.1 A Minimal Worked Trace

To make the apparatus concrete before the deterministic-core argument resumes, consider an abstract source S from which the proposer extracts two candidate facts p_1 and p_2 and proposes a candidate inference c whose conclusion q is anchored by a single bridge $b \in R$ instantiating a Walton-style scheme σ with critical

[48] Montague, R. (1970). Universal grammar. *Theoria*, 36(3), 373–398.

[49] Montague, R. (1973). The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik, & P. Suppes (Eds.), *Approaches to Natural Language* (pp. 221–242). Reidel.

[50] Kamp, H. (1981). A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, & M. Stokhof (Eds.), *Formal Methods in the Study of Language* (pp. 277–322). Mathematical Centre Tracts.

[51] Groenendijk, J., & Stokhof, M. (1991). Dynamic predicate logic. *Linguistics and Philosophy*, 14(1), 39–100.

[52] Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 658–666.

[53] Bos, J. (2008). Wide-coverage semantic analysis with Boxer. In *Proceedings of STEP 2008: Semantics in Text Processing*, 277–286. College Publications.

questions $\{cq_1, cq_2\}$. **Compile case.** Both p_1 and p_2 pass canonical-reference admission into F; b types correctly against σ; cq_1 is discharged by an admitted fact, cq_2 by a declared burden allocation accepted at the intermediate-graph boundary; the support DAG (leaves $p_1, p_2 \rightarrow$ bridge $b \rightarrow$ root q) is acyclic; $K(c) = a$, signed and emitted. **Refusal case** (near-identical candidate c' in which S omits the sub-claim grounding p_2): p_1 admits, p_2 does not, so the leaf required by b is absent; $K(c') = \perp_{\text{missing-fact}}$ with the missing leaf named. The contrast is the load-bearing property: the only structural difference between c and c' is one admitted fact, and that single difference is what the kernel localizes the refusal to. A reviewer who disputes the refusal is forced to dispute either the admission rule for p_2 or the bridge typing of b — both named, typed objects rather than an opaque verdict. A second property worth naming explicitly is *replay determinism*: the verdict $K(c)$ is a deterministic function of the candidate c, the Fact Register snapshot, the Catalog Registry snapshot, the deployment-tier rule, and the decision time, so that re-execution against the same pinned inputs reproduces the same compiled artifact (or the same refusal) bit-for-bit — the property an evidentiary regime requires to admit a compiled artifact as a reproducible record. The fully expanded trace, including the unanchored-bridge and unmet-critical-question refusal cases, is given in Appendix A.

3. The Deterministic Core

Cyc had the right architecture and the wrong era. Doug Lenat spent thirty years building what machine reasoning actually requires — a deterministic kernel operating on typed predicates over a curated knowledge base [54] [55] — and was treated as a punchline by every successive wave of statistical AI for being slow, brittle, and unable to scale. Cyc's *empirical* trajectory is contested: critics point to coverage gaps, the cost of hand-curated ontologies, and the absence of a commercial breakout as evidence that the project failed. We grant the empirical critique and reject the architectural inference. The mainstream consensus settled on the conclusion that GOFAI [56] failed *because the architecture was wrong*. The mainstream consensus was confusing the bottleneck with the architecture. The architecture was structurally correct for typed deterministic inference; the bottleneck was knowledge acquisition. Cyc lost the era it was built in because that era could not feed it; the architecture itself was never refuted by the evidence cited against it.

For forty years, every deterministic-kernel project — Cyc, TRIPS [57], SHRDLU [58], the entire Lisp-machine era — was throttled by the same problem: typed predicates require clean, structured inputs, and the world produces dense, ambiguous prose. The cost of converting prose into the kernel's input format was so high that the kernel's correctness was operationally irrelevant. You could prove a deterministic kernel sound and complete, and it would still lose to a statistical system that ingested raw text at a thousand times the speed.

The deep learning era resolved the bottleneck and drew the wrong conclusion. The capability that LLMs unlocked is not reasoning. It is *high-bandwidth extraction* — the ability to read dense, ambiguous prose and produce typed, structured candidate outputs at industrial scale. This is the capability Cyc spent forty years waiting for. The mainstream interpretation — that the LLM is the reasoner and the deterministic kernel is obsolete — inverts the architectural roles this paper recommends. Within the architecture proposed here, the LLM is the data-entry clerk that GOFAI (good old-fashioned AI, the symbolic-reasoning paradigm of pre-deep-learning AI) never had, and the kernel is the reasoner that GOFAI always was. We do not claim the mainstream interpretation is incoherent on its own terms; we claim that for the regulated-inference problem this paper scopes, the role assignment we propose is the better fit, for reasons §§4–6 develop.

[54] Lenat, D. B., & Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

[55] Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), 33–38.

[56] Haugeland, J. (1985). *Artificial Intelligence: The Very Idea*. MIT Press.

[57] Allen, J. F., Swift, M., & de Beaumont, W. (2008). Deep semantic analysis of text. In *Symposium on Semantics in Systems for Text Processing (STEP 2008)*. College Publications.

[58] Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1), 1–191.

This is the inversion the framework operates on. In this architecture, the frontier model is not treated as the intelligence layer; it is positioned as the ingestion layer. Its job is to read prose and propose typed structure: *here is a candidate fact, here is a candidate inference, here is a candidate bridge between them.* Every proposal is untrusted. The kernel admits or rejects. The LLM does at industrial scale what Cyc's human knowledge engineers did one predicate at a time, and the kernel that Lenat built — typed, deterministic, sound — gets to do the work it was always meant to do.

There is a loose structural parallel worth naming carefully. Chomsky's 1959 review of Skinner [59] refused the reduction of language to stimulus-response on the grounds that internal generative structure is what produces linguistic competence; the input-output trace is the surface, not the engine. The point of analogy here is not that retrieval-augmented generation [60] is Skinnerian behaviorism — RAG explicitly combines parametric and non-parametric memory and is engineered around the model's internal representations, not around stimulus-response — but that the *governance regime* layered on top of RAG-style systems often treats the retrieval-trace and the generated-token-stream as if they were the audit object. They are not. The model's weights produce the surface; the warrant for the surface is not in the trace. The framework's structural move is to make the warrant a separate, typed, inspectable object compiled by the kernel, so that audit operates on the warrant rather than on the trace. The Chomsky/Skinner parallel is illustrative of the *category mismatch* (trace vs. engine) and is not a claim that any specific contemporary AI system reproduces Skinner's theoretical commitments.

Wolfram is the closest commercial existence proof, and the analogy must be drawn carefully. Forty years of *Mathematica* [61] — typed symbolic computation operating deterministically on structured inputs — is an existence proof that a typed deterministic kernel operating on structured inputs *can* scale as a product rather than as a research artifact. The domain is mathematical symbolic computation, and the inputs are well-typed expressions in formal notation; the framework's domain is regulatory and evidentiary inference over admitted facts, and the inputs are typed candidate inferences over natural-language source material. The two are not the same problem and the engineering challenges do not transfer line-for-line. What does transfer is the architectural shape: a deterministic engine over typed inputs, shipped as production software, with refusal as a first-class output. Wolfram demonstrates the shape works at industrial scale in one domain. The framework's claim is that the same shape, with a different kernel and a different ingestion path, is the right paradigm for regulated inference. The mainstream ignored Wolfram's win because the inputs were inconvenient: someone had to type them by hand. The framework removes that constraint by replacing the human typist with the LLM proposer.

The combination — LLM proposer plus deterministic kernel plus structured-argument output — places the framework within the contemporary neurosymbolic landscape catalogued by Kautz [62]; specifically, it is closest to Kautz's "Neuro | Symbolic" pattern in which a neural component proposes and a symbolic component disposes, with the symbolic side holding the warrant. The architectural novelty here is not the pattern itself but the production-grade composition of the four lineages described above with a typed admission gate and tier-differentiated assurance contracts (§5).

What the compiler requires from this lineage is a deterministic engine that can execute structurally sound inferences over typed inputs, and a proposer that can produce those inputs at scale. Cyc and TRIPS built the engine. Wolfram proved it scales. The LLM, used correctly, supplies the inputs. The mainstream architectural account got the engine right and, for the regulated-inference problem this paper scopes, the role

[59] Chomsky, N. (1959). Review of B. F. Skinner's *Verbal Behavior*. *Language*, 35(1), 26–58.

[60] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

[61] Wolfram, S. (2003). *The Mathematica Book* (5th ed.). Wolfram Media.

[62] Kautz, H. (2022). The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. *AI Magazine*, 43(1), 105–125.

assigned to the LLM inverted.

A full account of the deterministic-core lineage would also engage the nonmonotonic-reasoning tradition (McCarthy's circumscription [63], Reiter's default logic [64]) and the probabilistic-knowledge-representation tradition (Pearl's Bayesian networks [65], Markov Logic Networks [66]), each of which addresses problems — defeasibility, uncertainty, partial information — that the symbolic-deterministic line did not natively solve. The framework's defeasibility machinery is handled at the argumentation layer (§4) rather than at the kernel's term language, and probabilistic reasoning is deliberately scoped out of the kernel's responsibilities; calibrated uncertainty over admitted facts is treated as a separate concern from compilation. These are scoping decisions, not denials that the literatures exist.

4. The Structural Audit

A compiler that produces correct output is not enough. The output has to survive review by an auditor who is not the compiler's author, was not present at production time, and may have a hostile interest in finding the failure mode. This is the requirement that argumentation theory was built to address, and the requirement that classical formal logic was specifically not built to address.

The foundational move belongs to Dung. The 1995 paper on abstract argumentation frameworks [67] established the field's mathematical substrate: arguments are nodes in a directed graph, an attack relation is the edge, and a battery of acceptance semantics (grounded, preferred, stable, complete) determine which arguments survive collectively. Every modern computational-argumentation system, including the ones the framework draws on, is downstream of Dung. Walton's argumentation schemes [68] [69] specialized Dung's abstract arguments into typed inference patterns, each with an explicit set of *critical questions* — the attacks the inference must survive to remain admitted. Prakken's ASPIC+ [70] [71] structured Walton's schemes into a defeasible-reasoning framework with strict and defeasible rules over a Dung-style attack graph. The Carneades tradition [72] specialized further into burden-of-proof allocation. A statistical inference is a different Walton scheme than a causal hypothesis is a different scheme than a legal classification, and each one declares its own critical questions. The framework's bridge typology is a Walton-style argumentation-scheme inventory; the framework's attack and defeat machinery is Dung–Prakken; every bridge type is a scheme, every scheme has critical questions, and every conclusion is admitted only if its supporting bridges have discharged those questions — either by answering them against admitted facts or, in intermediate non-emitted graphs, by formally allocating the burden in the Carneades sense; regulated-output artifacts (§5) require full discharge against admitted facts. Two graph structures must be distinguished and are routinely conflated: the *support* (provenance) DAG, which runs from admitted facts at the leaves through bridge edges to the conclusion at the root, is acyclic by construction — a cycle would mean a conclusion supports its own premise; the *attack/defeat* graph in the Dung sense, over which acceptance semantics are evaluated, may contain cycles

[63] McCarthy, J. (1980). Circumscription — A form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2), 27–39.

[64] Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1–2), 81–132.

[65] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[66] Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1–2), 107–136.

[67] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2), 321–357.

[68] Walton, D. (1996). *Argumentation Schemes for Presumptive Reasoning*. Erlbaum.

[69] Walton, D., Reed, C., & Macagno, F. (2008). *Argumentation Schemes*. Cambridge University Press.

[70] Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2), 93–124.

[71] Modgil, S., & Prakken, H. (2014). The ASPIC+ framework for structured argumentation: A tutorial. *Argument and Computation*, 5(1), 31–62.

[72] Gordon, T. F., Prakken, H., & Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15), 875–896.

and is handled by standard Dung machinery (grounded, preferred, stable, complete extensions). The kernel's acyclicity refusal applies to the support DAG; cyclic attack structure among candidate arguments is resolved by acceptance semantics, not refused. Computational implementations exist at research-prototype scale [73] [74]; the framework's contribution is not the formal model but the production-grade kernel built against it.

Bloom's taxonomy [75] [76] supplies the cognitive vocabulary the audit requires. The *Taxonomy of Educational Objectives* classified cognitive operations — recognition, comprehension, application, analysis, synthesis, evaluation — and treated each as a different kind of cognitive work with different mastery requirements. Applied to inference, the taxonomy answers a question Prakken alone does not: *what kind of move is the model attempting?* A bridge that recites an admitted fact is performing a different cognitive operation than a bridge that synthesizes a new conclusion across domains. They have different burdens. They have different failure modes. Treating them as a single category — *the model reasoned* — is what allows an unsupported synthesis to ride along with an admitted recitation, and is exactly the conflation an auditor needs to detect. The use of Bloom as a *typing discipline for machine inference* (rather than for human cognitive assessment, its original domain) is, to the present authors' knowledge, novel; a survey of the adjacent literatures supports the hedge. Bloom is widely deployed as a *learning-objective and assessment framework* in computing education — for course-outcome design, exam-item classification, and curriculum alignment in CS and software-engineering pedagogy [77] [78] [79] — and as a *content-difficulty rubric* for intelligent-tutoring-system item banks. It has been used to *grade LLM outputs* (classifying a model's answer to a question as recall versus analysis versus synthesis) in evaluation-benchmark work; it has not, on the present authors' reading, been used as a *type system on the inference object itself*, in which each bridge in a compiled inference graph carries a Bloom-level tag that the kernel checks for compatibility with the bridge's argumentation scheme and that the auditor reads as a load-bearing typing constraint. The cited tradition treats Bloom as a *vocabulary applied to humans or to outputs*; the framework treats Bloom as a *type system on a typed inference graph*. The distinction is the operative novelty.

Bloom and Prakken pair tightly. Bloom names the cognitive operation; Prakken supplies the scheme that operation must satisfy; the framework's typed transform is the implementation. A bridge is a Bloom-classified cognitive operation executed as a Prakken scheme with declared critical questions and a typed input-output signature. The kernel checks the signature. The auditor checks the scheme. The cognitive type is the vocabulary they share.

Reed and Kellogg supplied the original visualization, almost 150 years before either Bloom or Prakken wrote a word. The 1877 sentence diagram [80] solved a problem few modern formalisms solve as cleanly: making structural failure visible to a non-expert reader. Subject and predicate on a horizontal baseline. Modifiers on diagonals, anchored to what they modify. Every word has exactly one structural home. A floating modifier is not a stylistic flaw or a near-miss; it is a diagrammatic error visible at a glance, even to a reader who could not name the grammatical rule it violates.

[73] Gordon, T. F., Prakken, H., & Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15), 875–896.

[74] Modgil, S., & Prakken, H. (2014). The ASPIC+ framework for structured argumentation: A tutorial. *Argument and Computation*, 5(1), 31–62.

[75] Bloom, B. S. (Ed.). (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. David McKay.

[76] Krathwohl, D. R. (2002). A revision of Bloom's Taxonomy: An overview. *Theory into Practice*, 41(4), 212–218.

[77] Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for CS assessment. In *Proceedings of the Tenth Conference on Australasian Computing Education* (pp. 155–161). Australian Computer Society.

[78] Oliver, D., Dobele, T., Greber, M., & Roberts, T. (2004). This course has a Bloom rating of 3.9. In *Proceedings of the Sixth Australasian Conference on Computing Education* (pp. 227–231). Australian Computer Society.

[79] Johnson, C. G., & Fuller, U. (2012). Is Bloom's taxonomy appropriate for computer science? In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research* (pp. 120–123). ACM.

[80] Reed, A., & Kellogg, B. (1877). *Higher Lessons in English: A Work on English Grammar and Composition*. Clark & Maynard.

Reed-Kellogg is the historical ancestor; the modern technical lineage descends from it through dependency grammar [81] [82], head-driven phrase structure grammar [83], and the dependency-parsing tradition that produced Universal Dependencies [84] and the contemporary semantic-graph formalisms — Abstract Meaning Representation [85], Universal Conceptual Cognitive Annotation [86], and the bank of typed-edge resources used by modern computational linguistics. The mature versions are richer than the 1877 original in every technical dimension: they handle long-distance dependencies, coordination, ellipsis, and cross-linguistic variation that Reed-Kellogg cannot. AMR's design specification, in particular, explicitly seeks rooted, labeled graphs that are "easy for people to read" [87], so the legibility property is not abandoned in modern formalisms so much as differently constrained — toward annotator-readability for trained linguists rather than toward audit-readability for untrained regulators. The framework draws on the modern formalisms for its typed-edge inventory and its handling of non-tree structures; what it imports from Reed-Kellogg specifically is the audit-readability variant of the legibility property: structural failure visible at a glance to a reader who has not been trained on the formalism. A dependency parse provides a formally constrained syntactic structure; an AMR graph is expressive over predicate-argument structure; both reward training in their respective notations. The framework's audit interface is a deliberate hybrid: the structural rigor comes from the modern lineage, the audit-readability constraint comes from Reed-Kellogg's specific solution to the *no-prior-training* legibility problem.

This is the property the framework imports from each side. The extractor produces a typed directed acyclic graph that inherits the modern lineage's structural rigor — typed edges with declared semantics, non-tree structures where the inference requires them, principled treatment of cross-references — and generalizes it under the Reed-Kellogg readability constraint: every load-bearing bridge must anchor to an admitted fact, transitively, or the compile fails, and the resulting graph must be readable by a non-expert auditor without specialized training. The diagram is not a UI rendering of the kernel's state. It *is* the kernel's state, projected into a form a non-expert auditor can read.

Acyclicity is enforced at compile time, not at render time: a cycle in the inference graph would mean a conclusion supports its own premise, and the kernel refuses such structures as the formal expression of hidden-premise refusal — the *petitio principii* that classical logic names but cannot, on its own, prevent. Nodes carry Bloom-classified cognitive-type tags; edges carry Prakken argumentation schemes with their critical questions; the auditor reads the graph in topological order from admitted facts at the leaves to the conclusion at the root. The DAG is, in this sense, the load-bearing data structure of the entire architecture: it is what the Fact Register feeds, what the kernel compiles, what the audit interface renders, and what the signing layer attests. Every other component in the system exists to produce, evaluate, display, or certify it.

The pairing with Prakken is where the audit becomes mathematically unforgiving. A floating diagonal in a Reed-Kellogg diagram and a Walton-style bridge with an undischarged critical question play the *same audit role*: each is a structurally locatable failure in a typed graph, each is visible without specialist training, and each refuses to compile. The two are structurally analogous, not literally the same object. The framework's interface does not show a regulator a *summary* of what the kernel concluded; it shows the regulator the

[81] Tesnière, L. (1959). *Éléments de syntaxe structurale*. Klincksieck. English translation: *Elements of Structural Syntax* (2015), trans. T. Osborne & S. Kahane, John Benjamins.

[82] Mel'čuk, I. A. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.

[83] Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

[84] Nivre, J., et al. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. *Proceedings of LREC 2016*, 1659–1666.

[85] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., et al. (2013). Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop*, 178–186.

[86] Abend, O., & Rappoport, A. (2013). Universal Conceptual Cognitive Annotation (UCCA). *Proceedings of ACL 2013*, 228–238.

[87] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., et al. (2013). Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop*, 178–186.

kernel's compiled state, in a notation that a nineteenth-century schoolchild could read. When a bridge fails, the failure is visible diagrammatically, classifiable cognitively, and defensible argumentatively. The auditor does not need to trust the framework. The auditor reads the diagram.

What the compiler requires from this lineage is an output that survives hostile review. Prakken makes the bridges challengeable. Bloom makes the challenges typed. Reed-Kellogg makes the resulting structure legible to a reader without a doctorate. Without all three, the compiler produces output that is correct and unauditable, which in regulated environments is functionally the same as wrong.

5. Differentiated Assurance

A compiler that produces correct, auditable output still has to be deployed somewhere, and the somewhere matters. A research environment tolerates uncertainty that a regulated bank does not. A regulated bank tolerates latency that a classified intelligence facility does not. A classified facility tolerates unavailability that a research environment does not. The framework cannot ship one assurance posture and call it general; it has to ship a structured argument, environment by environment, for what kind of failure each environment will and will not tolerate.

This is the requirement that the Goal Structuring Notation tradition was built to address. Tim Kelly's 1998 thesis at York [88] formalized the *assurance case* as a tree of claims, evidence, and reasoning that defends a top-level safety goal against the specific threat model of a specific deployment, and the GSN community standard [89] consolidates the notation in current use. GSN sits within a broader assurance-case ecosystem: ISO/IEC 15026 [90] standardizes the assurance-case concept across systems engineering; aviation software certification under DO-178C [91] formalizes objective-based assurance for safety-critical software, with adjacent guidance for tool qualification (DO-330) and model-based development (DO-331); the AMLAS methodology [92] specializes assurance-case discipline to machine learning components in autonomous systems. Different environments require different cases. Aviation cannot ship the assurance posture of consumer software. Nuclear cannot ship the assurance posture of aviation. The discipline these standards supply is not a checklist; it is the structured-argument grammar for proving, environment by environment, that a system's failure modes have been enumerated and that the residual risk is acceptable to that environment's stakeholders. Recent work has begun applying assurance-case discipline to AI systems specifically [93] [94], but no production framework has shipped tier-differentiated assurance as an architectural feature.

The framework's deployment tiers are GSN-formalized failure-mode contracts, not product SKUs.

Studio fails *soft*. The customer's model is in the loop, governance is lighter, the kernel admits more provisional bridges, and the failure mode the environment tolerates is *the wrong answer reaches the user, who*

[88] Kelly, T. P. (1998). *Arguing safety: A systematic approach to managing safety cases* (Doctoral dissertation). University of York, Department of Computer Science.

[89] Assurance Case Working Group. (2021). *Goal Structuring Notation Community Standard, Version 3*. Safety-Critical Systems Club.

[90] International Organization for Standardization. (2019). *ISO/IEC 15026-2:2019 — Systems and software engineering — Systems and software assurance — Part 2: Assurance case*. ISO.

[91] RTCA. (2011). *DO-178C: Software Considerations in Airborne Systems and Equipment Certification*. Radio Technical Commission for Aeronautics.

[92] Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., & Habli, I. (2021). Guidance on the assurance of machine learning in autonomous systems (AMLAS). *University of York Assuring Autonomy International Programme*.

[93] Bloomfield, R., & Rushby, J. (2021). Assurance 2.0: A manifesto. In *Proceedings of the 29th Safety-Critical Systems Symposium*.

[94] Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., & Habli, I. (2021). Guidance on the assurance of machine learning in autonomous systems (AMLAS). *University of York Assuring Autonomy International Programme*.

is expected to evaluate it. This is the assurance posture appropriate to exploration, model evaluation, and pre-production analysis.

Refinery fails *closed*. The Goober runtime (a domain-fine-tuned model on an open base) is in the loop, full bridge typing is required, and the failure mode the environment tolerates is *the system declines to answer when it cannot compile.* This is the assurance posture appropriate to regulated deployment — Fair Lending, credit decisioning, eligibility determination, regulated medical inference. The kernel's silence is a feature; the worst failure mode is an unsupported answer reaching a regulated decision.

Clean Room fails *hard*. Cryptographic attestation is required, the deployment posture is zero-egress behind a data diode, and the failure mode the environment tolerates is *the system halts before it can leak state.* This is the assurance posture appropriate to classified facilities, sensitive intelligence work, and any environment where exfiltration is a worse failure than unavailability.

Same compiler. Three failure-mode contracts. The kernel's correctness is invariant across tiers; what varies is the environment's definition of acceptable failure and the structured argument that defends the deployment to that environment's stakeholders. A Refinery deployment cannot be retrofitted from a Studio deployment by adding governance, because the assurance argument for Refinery is structurally different from the assurance argument for Studio. The framework is built so each tier compiles its own assurance case from the ground up.

A worked GSN tree per tier — with explicit top-level goals, strategies, contexts, and evidence nodes — is engineering work downstream of the architectural commitment and is out of scope for this paper. The position taken here is that *the differentiated-tier model is the correct architectural granularity for assurance-case authoring in regulated AI*, and that this granularity is what the existing AI-assurance literature has yet to ship as a product feature. Concrete tree authoring against ISO 15026 and DO-178C-style objective decomposition does not change the architectural argument either way.

What the compiler requires from this lineage is the discipline to refuse one-size-fits-all deployment. GSN supplies the grammar. The framework's tiers are the grammar made operational. A regulator reading the Refinery assurance case is reading a structured argument that has been compiled for them, in their environment, against their threat model. They are not reading marketing.

6. Integration

Each lineage supplies one thing the compiler requires. Frege supplies canonical reference identity. Tarski supplies the model-relative satisfaction relation. Montague supplies the prose-to-structure bridge. Lenat supplies the kernel architecture. Allen and the TRIPS tradition supply the proof that deterministic logic works on clean inputs. Wolfram supplies the closest commercial analogue — typed deterministic computation shipped as a production product in the mathematical-symbolic-computation domain — from which the architectural shape (not the domain transfer) is borrowed. Chomsky supplies the *illustrative* construction-not-trace analogy clarified in §3, not a load-bearing theoretical commitment. Dung, Walton, and Prakken supply defeasible inference and acceptance semantics. Bloom supplies the cognitive taxonomy, novel-in-application as a type system for machine inference. Reed and Kellogg supply the audit-readability constraint. Deming supplies the quality philosophy. Kelly and the GSN tradition, with ISO/IEC 15026, DO-178C, and AMLAS, supply the differentiated assurance posture.

Figure 1 shows the integration as a runtime architecture. The frontier model is structurally upstream of the kernel and produces typed candidates only; the kernel admits or rejects against the closed Fact Register and the typed Catalog Registry; the compiled output is rendered diagrammatically for audit and emitted as a signed artifact suitable for the deployment tier's assurance case.

Figure 1. *Runtime architecture: lineage debts mapped onto runtime components.* The frontier model is bracketed on both sides — upstream by the catalog-driven retrieval orchestrator (which determines what the model can see), downstream by the deterministic kernel (which determines what the model's output can become). Lineage attributions in italics: *Cyc/Wolfram inversion* on the LLM proposer; *Frege canonical-reference identity, Tarski satisfaction* on the Fact Register; *Walton/Dung–Prakken* on the Catalog Registry; *Lenat/Allen, construction-not-trace* on the kernel; *Bloom + Reed-Kellogg* on the audit interface; *Kelly/GSN, ISO 15026, DO-178C, AMLAS* on the deployment tiers.

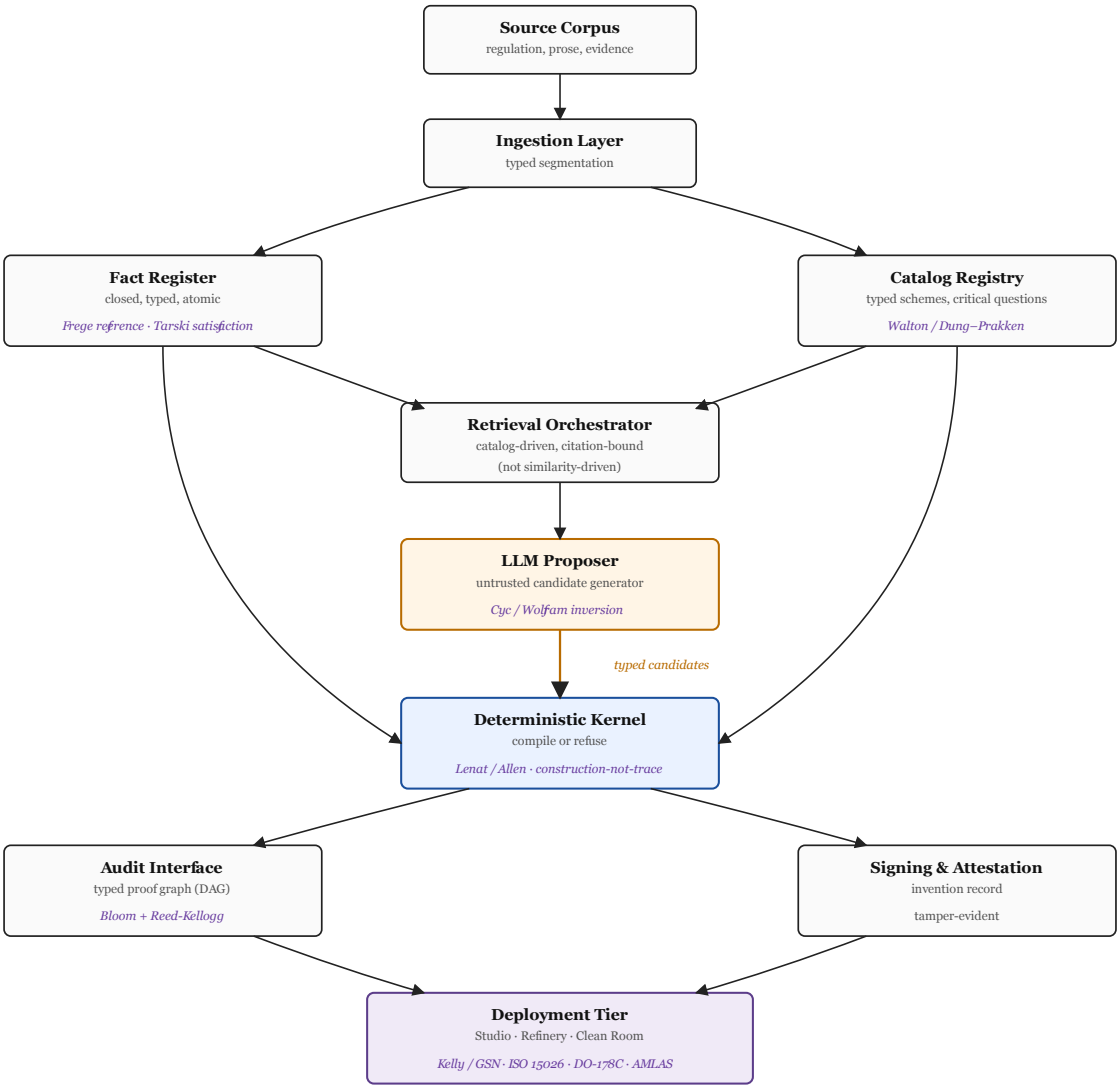

Figure 1: Figure 1: Runtime architecture mapping intellectual lineages onto runtime components.

The diagram makes one structural property visible: the LLM is bracketed on both sides. Upstream, the retrieval orchestrator constrains what the LLM can see by retrieving load-bearing provisions from the Catalog Registry by citation rather than by similarity, inverting standard retrieval-augmented generation [95]. Downstream, the kernel constrains what the LLM's output can become by refusing to compile any bridge that does

[95] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

not anchor to admitted facts. The model is free to be a model — high-bandwidth, probabilistic, fluent — between two surfaces that are neither.

Each thread, alone, has practitioners. Linguists work in generative grammar. Educators work in Bloom. Verification engineers work in formal methods. Quality consultants work in Deming. Safety engineers work in GSN. The combination — someone holding all of them at once, with the architectural instinct to fuse them into a single deployable runtime — does not exist as a discipline. It exists as a framework, and the framework is what produces the compiler.

6.1 Why Automated Evaluation Does Not Close the Inspection Gap

The strongest live counter-argument to the inspection-collapse thesis in §1 is that LLM-as-judge automated evaluation [96] [97] [98] reduces the marginal cost of grading enough that inspection no longer scales linearly with deployment volume. If a judge model can score a response in milliseconds at the cost of a single forward pass, the argument runs, then inspection is asymptotically cheap and the structural-construction discipline of this paper is solving a problem automation has already eliminated. The objection is serious and the response is structural.

First, automated judges *grade*; they do not *construct*. A judge model returns a verdict on a candidate output, but the verdict is itself a probabilistic guess over a fluent surface, with the same trace-vs-engine category mismatch (§3) as the system being judged. Treating one probabilistic surface as the audit of another does not introduce a load-bearing structural object into the loop; it introduces a second probabilistic surface whose failures correlate with the first one's because both share base-model lineage, training distribution, and instruction-following idioms. The empirical literature on judge bias — positional preference, self-preference, length bias, sycophantic agreement, and distributional drift between judge and judged — is now a recognized failure-mode catalog [99] [100], and each of these failures grows with deployment volume rather than amortizing over it. A judge that is itself an LLM does not break the inspection-collapse argument; it relocates it.

Second, even a perfectly calibrated judge does not produce the artifact a regulator needs. A regulator is not asking *did the model probably get this right*. A regulator is asking *show me the typed inference, show me the admitted facts it anchors to, show me the discharged burden of proof.* A judge score does not contain these objects. A compiled DAG does. The construction layer and the judge layer are not substitutes; they answer different questions for different audiences. A judge can be a useful component *inside* a runtime governance loop (§1.1) — for example, scoring whether an agent's tool-use trajectory looks suspicious — but it cannot replace a typed-inference kernel for the same reason a defect-detection camera at the end of an assembly line cannot replace the production-line discipline that prevents defects from being built.

Third, the cost argument cuts in the construction layer's favor at high deployment volume. A compiler's cost is paid once at compile time per inference, and the compiled artifact is reusable across audits, attestations, and downstream uses. A judge's cost is paid every time inspection is requested, and in regulated domains inspection is requested on every output that affects a regulated decision. The crossover happens earlier than the judge-as-replacement argument acknowledges: the construction layer pays a fixed admission cost, the inspection-with-judge layer pays a per-decision cost that grows with deployment volume and re-accrues

[96] Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36.

[97] Gu, J., Jiang, X., Shi, Z., et al. (2024). A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*.

[98] Li, H., Dong, Q., Chen, J., et al. (2024). LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

[99] Gu, J., Jiang, X., Shi, Z., et al. (2024). A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*.

[100] Li, H., Dong, Q., Chen, J., et al. (2024). LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

every time the threat model changes or the judge is updated. Goodhart's law [101] [102] applies with particular force to optimized judges: when the judge becomes the target metric, the system upstream learns to satisfy the judge rather than the underlying property the judge was meant to measure, and the construction layer is what prevents that loop from closing.

The position taken here is that automated evaluation is a useful runtime-governance tool that composes with the construction layer rather than replacing it, and that the absence of either layer is the gap the present literature has not yet closed.

6.1.1 A Cost-Crossover Sketch

The scaling argument can be made precise as a small comparative cost model. Let N be the deployment volume (regulated inferences per period) and let u be the number of distinct downstream uses per inference (audit, attestation, regulator response, internal review, customer disclosure). Inspection cost under an LLM-as-judge regime is $C_{\text{insp}}(N, u) = N \cdot u \cdot c_j + N \cdot u \cdot c_h \cdot \eta$, where c_j is the per-call cost of the judge, c_h is the per-incident cost of human escalation, and $\eta \in [0, 1]$ is the rate at which judge scores are escalated to a human reviewer. The judge cost is paid every time inspection is requested, and re-accrues whenever the threat model changes or the judge is updated.

Construction cost under the kernel regime is $C_{\text{cons}}(N, u) = N \cdot c_k + F$, where c_k is the per-inference admission cost (LLM proposal plus kernel evaluation) and F is the fixed amortizable cost of authoring the Catalog Registry and Fact Register schemas for the regulated domain. The compiled artifact is reusable across the u downstream uses without re-incurring per-use cost. The crossover condition $C_{\text{cons}}(N, u) < C_{\text{insp}}(N, u)$ resolves to $F < N \cdot (u \cdot (c_j + c_h\eta) - c_k)$, which is satisfied whenever the per-decision inspection cost $u \cdot (c_j + c_h\eta)$ exceeds the per-decision admission cost c_k by enough to amortize the fixed catalog cost F over the deployment volume N. Three properties of the inequality matter for the architectural argument: (i) it is *linear in* N on both sides, but the construction side has a fixed offset F and the inspection side does not, so the crossover is a function of catalog amortization rather than of any super-linear scaling claim; (ii) it is *multiplicative in* u on the inspection side and *constant in* u on the construction side, so use-multiplicity strictly favors the construction layer in regulated domains where the same inference must be defensible to multiple distinct audiences; and (iii) it is *insensitive to threat-model updates* on the construction side and *sensitive on the inspection side*, because re-running compiled artifacts under a new judge is free while re-running judge inspection under a new judge is not.

A back-of-envelope calibration makes the crossover concrete. Take plausible mid-2026 parameter values for a regulated-decision pipeline: judge cost $c_j = \$0.002$ per call (a single forward pass on a small judge model), admission cost $c_k = \$0.01$ per inference (LLM proposer plus kernel evaluation, dominated by the proposer), human-escalation cost $c_h = \$50$ per incident (a compliance-analyst minute, fully loaded), escalation rate $\eta = 0.05$ (one in twenty judge scores escalated), downstream-use multiplicity $u = 3$ (audit, regulator response, internal review), and fixed catalog-build cost $F = \$150{,}000$ (legal-and-engineering authoring of one regulated regime's Catalog Registry and Fact Register schemas). Substituting, the per-decision inspection cost is $u \cdot (c_j + c_h\eta) = 3 \cdot (\$0.002 + \$2.50) = \$7.506$ and the per-decision admission cost is $c_k = \$0.01$, so the crossover volume is $N^* = F/(\$7.506 - \$0.01) \approx 20{,}010$ regulated inferences over the catalog's amortization period. Sensitivity is dominated by the human-escalation term: holding everything else fixed, halving η to 0.025 raises N^* to $\approx$ 40,030; doubling u to 6 lowers N^* to $\approx$ 9,995; tripling F to \$450,000 raises N^* to $\approx$ 60,030. Across this range the crossover lands between 10^4 and

[101] Goodhart, C. A. E. (1975). Problems of monetary management: The U.K. experience. In *Papers in Monetary Economics*. Reserve Bank of Australia.

[102] Manheim, D., & Garrabrant, S. (2019). Categorizing variants of Goodhart's Law. *arXiv preprint arXiv:1803.04585*.

10^5 regulated inferences — well below the per-month volumes a single mid-sized regulated-AI deployment produces, and orders of magnitude below the frontier-model regime. The numeric values are illustrative and not calibrated against a deployed system; the structural point is that the crossover exists and falls inside the range a regulated deployment routinely operates in, not that any particular cell of the table is correct.

7. Falsifiability and Forthcoming Work

The argument in this paper is architectural, not empirical. It claims that an architecture combining the four lineages above is the right paradigm for machine reasoning in regulated domains. A claim of that shape is falsifiable in three ways, and the falsifiability conditions deserve to be stated explicitly.

First, the architecture is falsifiable at the kernel: the kernel's evaluation algorithm must be sound against its specification — every bridge it admits must in fact anchor to admitted facts, every conclusion it compiles must in fact discharge its declared burdens, every hidden premise must in fact be refused. The intended discipline is that each failing test localizes to one of three named layers — a specification error, an implementation error, or an ontology error in the Fact Register or Catalog Registry — rather than being patched in place. Operationalizing this condition with a specification-anchored test suite is engineering work the present authors intend to pursue separately; the discipline does not depend on any particular implementation surviving review, only on the architectural claim that the three error layers are the correct decomposition.

Second, the architecture is falsifiable at the catalog: a deterministic regulation-text-to-catalog-stub pipeline must produce, for at least one full regulatory regime, a catalog whose load-bearing provisions match a hand-authored reference catalog within tolerable bounds. This is a discharge surface for the bracketed-LLM pattern (§6) at its upstream boundary; whether any particular extraction pipeline succeeds is independent of the architectural claim that catalog-driven retrieval is the correct upstream bracket.

Third, the architecture is falsifiable at the assurance case: the GSN-formalized failure-mode contracts for Studio, Refinery, and Clean Room must be defensible against hostile review by stakeholders in their respective environments. This is the one falsifiability condition that cannot be discharged by an in-house artifact; it requires deployment and external audit. The framework is calibrated to invite that audit rather than survive it.

The position of this paper is that all three conditions are dischargeable — that the architecture is correct in shape and that the implementation work, while substantial, is engineering rather than research. A reader who disagrees should be able to identify which of the three conditions they expect to fail, and why. That specificity of disagreement is itself a benefit of the architecture: there is no hand-wave layer at which the system can hide.

7.1 Limitations and Threats to Validity

The argument is bounded in ways that should be stated explicitly rather than inferred from the hedges.

Scope limitation: regulated inference, not general AI. The architectural claim is restricted to domains where (a) the inference is regulated or otherwise subject to structured audit, (b) the load-bearing premises are expressible as typed facts admissible into a closed register, and (c) the bridges between premises and conclusions are expressible as a finite catalog of argumentation schemes. Domains in which any of the three conditions fails — open-ended creative generation, unbounded common-sense reasoning, conversational interaction without an audit object — are out of scope. The paper does not claim that construction-time admission is the right paradigm for all uses of large language models; it claims it is the right paradigm for the regulated-inference subset.

Evidentiary limitation: architectural specification, not deployed evaluation. The contribution is a specification of K, a four-lineage decomposition, and a deployment-tier model. The paper does not include deployed-system evaluation, A/B comparison against inspection-paradigm baselines, or empirical measurement of the cost-crossover model in §6.1.1. A reader expecting an empirical paper will not find one. A reader expecting an architecture paper with explicit falsifiability conditions (§7) will.

Formal-properties limitation: stated, not proved. Admission soundness, refusal totality, and canonicalization determinism (§1.2) are stated as *structural properties of the construction* — properties the kernel is engineered to satisfy by virtue of its admission discipline, its refusal-locator coverage, and its canonicalization procedure — not as theorems with published mechanized proofs. Discharging each as a proof obligation against an executable specification is forthcoming work; the architectural argument here does not depend on any particular proof artifact existing, only on the claim that the three properties are the correct decomposition of what the kernel must guarantee.

Catalog-construction risk. The architecture's correctness is bounded by the correctness of the Catalog Registry R and the Fact Register F. A misauthored bridge in R admits inferences that should refuse; a misadmitted fact in F anchors inferences that should not be anchored. The architecture localizes such failures (each becomes a typed object an auditor can name and find), but it does not eliminate them. This is the standard risk surface for any typed-admission discipline and is shared with theorem provers, type systems, and policy engines; it is not unique to this architecture, but it is real, and consumers of the architecture should price it accordingly.

Proposer-coverage risk. The architecture assumes the LLM proposer can produce candidate facts and candidate bridges at industrial scale with sufficient recall that the kernel's refusals are informative rather than vacuous. If the proposer systematically fails to produce candidates the kernel would have admitted, the system silently under-emits compiled artifacts. The kernel's refusal totality (§1.2) protects against silent *acceptance* of bad inferences; it does not protect against silent *omission* of good ones. Proposer recall is an empirical property the architecture does not establish.

Semantic-extraction risk. The Frege/Tarski/Montague apparatus (§2) supplies the discipline that compositional truth is computable; modern semantic parsing supplies working extraction technology. Neither supplies a guarantee that any particular natural-language source can be reduced to a typed graph without loss of load-bearing content. Edge cases — metaphor, irony, presupposition under negation, scope ambiguity — remain hard problems in computational semantics and are not solved by this paper. The architecture handles such cases by refusing to admit candidates the proposer cannot type cleanly; the refusal is sound, but the under-coverage is real.

External-validity risk: the assurance-tier argument. The differentiated-assurance model (§5) claims that Studio/Refinery/Clean Room is the correct architectural granularity. This claim is falsifiable only by deployment under external audit (§7, third condition). Until that audit is conducted, the tier model is a proposal, not a validated finding. The position of this paper is that the proposal is structurally correct; readers are invited to disagree on the specific tier definitions or on the granularity itself.

Adversarial-proposer risk. The architecture treats the LLM as untrusted but not as adversarial. A proposer optimizing against the kernel's admission rules — producing candidate bridges crafted to type-check while smuggling unsupported conclusions through the critical-question discharge — is a threat model the present paper does not address. Goodhart-style pressure (§6.1) on the kernel's admission rules is a research direction rather than a solved problem; the construction layer reduces but does not eliminate adversarial surface.

Citation and disclosure. The reference to the present authors' separate work [103] is a pointer to ongoing

[103] Fishburn, S. (2026). *Technical foundations: Layered disclosure for a structural-verification AI architecture.* Kenshiki Labs.

technical work, not a load-bearing premise of the argument here; readers should treat it as supplementary rather than as an appeal to authority. The paper's conclusions stand or fall on the architectural argument as presented, not on the existence of any particular companion artifact.

8. Conclusion

The mainstream AI industry is solving a different problem with a different paradigm. They are building larger probabilistic models and grading their outputs with inspection regimes that scale linearly with deployment volume. The work is rigorous. The paradigm is wrong. The history of every prior industry that mistook inspection for quality — Detroit, then American steel, then American semiconductors — is the history of incumbents who held the wrong theory of where quality comes from and lost industries they had invented to competitors who held the right one. The pattern is structural, not accidental. It applies again here.

The architecture in this paper is what the right paradigm looks like for machine reasoning. Frege, Tarski, Montague, Lenat, Allen, Wolfram, Chomsky, Bloom, Dung, Prakken, Walton, Reed, Kellogg, Deming, Kelly. Each name is a debt. The compiler is what the debts compose into. The invention record is what the compiler emits. If regulated AI shifts toward structurally compiled inference, architectures that build quality into the construction of an inference rather than into the inspection of its surface will be advantaged — and the case for that shift is what this paper has tried to make falsifiable rather than merely asserted.

References

Appendix A. Worked Trace: One Compile, Three Refusals

This appendix expands the §2.1 vignette into a full end-to-end trace. The example is deliberately abstract — predicates are named p_i, bridges b_j, schemes σ_k — so that the structural discipline rather than any domain-specific accident is what the reader audits. The trace is not a transcript of any deployed system; it is the smallest example sufficient to exhibit the four kernel outcomes (one compile and three structurally distinct refusals) over a single Fact Register F and Catalog Registry R.

A.1 Setup

Let the Catalog Registry R contain one bridge type:

- $b : \sigma$, instantiating a Walton-style argumentation scheme σ whose declared signature is $\sigma(p_1, p_2) \vdash q$ — that is, b admits a conclusion q when premises p_1 and p_2 are both admitted into F. The scheme σ declares two critical questions: cq_1 (*are p_1 and p_2 jointly applicable to the entity referred to by q?*) and cq_2 (*is the entity referred to by q within the scope σ governs?*).

Let F be initially empty. Let the canonical-reference identity relation enforced at admission be $\equiv_F$: two surface mentions admit to the same fact iff they resolve to the same registered referent under the Fregean discipline of §2.

Four candidate inferences are considered, each derived by the LLM proposer from a slightly different source text S_i. All four candidates target the same conclusion q via the same bridge b. The kernel processes each independently.

Companion working paper, in preparation, targeted for SSRN posting in Q3 2026.

A.2 Case 1 — Compile success

Source S_1. Contains assertions whose extraction yields candidate facts $\tilde{p}_1, \tilde{p}_2$, each carrying a referent label ρ_1, ρ_2 resolved by the proposer.

Admission. The Fact Register checks ρ_1, ρ_2 against registered referents under $\equiv_F$. Both resolve. p_1 and p_2 are admitted into F.

Bridge typing. Candidate $c_1 = (b; p_1, p_2; q)$ is presented. The kernel checks the signature of b against σ: b's declared input types match p_1, p_2; output type matches q. Typing succeeds.

Critical questions. cq_1 is discharged by an admitted fact $p_3 \in F$ asserting joint applicability. cq_2 is discharged by a declared burden allocation under the Carneades discipline, formally accepted at the intermediate-graph boundary; the resulting graph is then promoted to a regulated-output artifact only after cq_2 is re-discharged against an admitted scope-fact $p_4 \in F$ (per §1.2 and §4: regulated-output artifacts require all critical questions answered against admitted facts).

Acyclicity. The support DAG has leaves $\{p_1, p_2, p_3, p_4\}$, one bridge node b, and root q. No cycles.

Outcome. $K(c_1) = a_1$. The artifact a_1 is canonicalized (lexicographic node ordering, scheme-instantiation normalization, provenance-edge deduplication), signed, and emitted as a compiled artifact suitable for the deployment tier's assurance case.

A.3 Case 2 — Refusal $\perp_{\text{missing-fact}}$

Source S_2. Identical to S_1 except the sub-claim grounding p_2 is absent from the source text.

Admission. Extraction yields $\tilde{p}_1$ only; no candidate $\tilde{p}_2$ is produced because no source assertion grounds it. p_1 is admitted into F. p_2 is not in F.

Bridge typing. Candidate $c_2 = (b; p_1, p_2; q)$ is presented. The kernel checks the leaves required by b's signature: $p_1 \in F, p_2 \notin F$.

Outcome. $K(c_2) = \perp_{\text{missing-fact}}$, with the missing leaf p_2 named. The refusal is structurally locatable: the failure is at the Fact Register, not at the bridge or the scheme. An auditor who disputes the refusal must argue either that p_2 should have been admitted (a failure of the admission rule applied to S_2) or that b's signature is wrong (a failure in R). Both are typed, named objects.

A.4 Case 3 — Refusal $\perp_{\text{unanchored-bridge}}$

Source S_3. Contains assertions grounding both p_1 and p_2, but the surface mention intended to anchor p_2 uses a referent label ρ_2' that does not resolve to any registered referent under $\equiv_F$ — for example, an ambiguous moniker for which the Fregean sense/reference discipline cannot select a unique referent without further evidence.

Admission. p_1 admits. The candidate $\tilde{p}_2$ fails canonical-reference admission: ρ_2' does not collapse to a registered referent. $\tilde{p}_2$ is rejected at admission. The proposer, lacking p_2 in F, instead proposes a substitute bridge $b' \in R$ that attempts to anchor q via p_1 and a directly extracted surface predicate $\tilde{p}_2'$ that the proposer has *not* routed through Fact Register admission.

Bridge typing. Candidate $c_3 = (b'; p_1, \tilde{p}_2'; q)$ is presented. The kernel checks every leaf required by b' against F: $p_1 \in F; \tilde{p}_2' \notin F$. The bridge has a leaf that is not anchored to an admitted fact.

Outcome. $K(c_3) = \perp_{\text{unanchored-bridge}}$, with the unanchored leaf $\tilde{p}_2'$ and the responsible bridge b' both named. This refusal is distinct from Case 2: in Case 2 the bridge was correctly typed against σ but a required fact was absent from the source; in Case 3 the source contains a surface assertion but the assertion fails canonical-reference admission and the bridge attempts to route around the Fact Register. The refusal localizes to the *bridge–register interface*, which is precisely the failure mode the §1.1 discussion of "hidden premises" names.

A.5 Case 4 — Refusal $\perp_{\text{unmet-critical-question}}$

Source S_4. Identical to S_1 in admitted facts: p_1, p_2, p_3 all admit into F. The scope-fact p_4 that discharged cq_2 in Case 1, however, is not present in S_4 and is not derivable from any admitted fact.

Bridge typing. Candidate $c_4 = (b; p_1, p_2; q)$ types correctly against σ. Leaves admit.

Critical questions. cq_1 discharges against p_3. cq_2 has no admitted fact answering it. The proposer attempts to mark cq_2 as carried forward under a Carneades burden allocation — admissible at the intermediate-graph boundary — and then to promote the resulting graph directly to a regulated-output artifact.

Outcome. $K(c_4) = \perp_{\text{unmet-critical-question}}$, with cq_2 named as the undischarged question and σ named as the scheme that requires it. The intermediate graph would have been admissible (per §1.2 and §4); the regulated-output artifact is not, because the regulated-output rule requires all critical questions to be answered against admitted facts. This refusal is distinct from Cases 2 and 3: facts were admitted, the bridge was anchored, the support DAG was acyclic — the failure is at the *argumentation layer*, in the discharge of a typed critical question against the deployment tier's assurance-case requirements.

A.6 What the Trace Demonstrates

The four cases share a single bridge $b : \sigma$, a single conclusion q, and near-identical source texts. They differ in exactly which structural condition fails. The kernel's refusals are not graded scores or confidence intervals; each is a typed object naming a specific structural failure at a specific layer:

Case	Outcome	Failure layer	Named object
1	$K(c_1) = a_1$	—	(compiled artifact)
2	$\perp_{\text{missing-fact}}$	Fact Register	required leaf p_2
3	$\perp_{\text{unanchored-bridge}}$	Bridge–register interface	unanchored leaf $\tilde{p}_2'$, bridge b'
4	$\perp_{\text{unmet-critical-question}}$	Argumentation layer	critical question cq_2, scheme σ

The property the trace exhibits is the one §1.2 specifies as *refusal totality*: every input either compiles or refuses with a structurally locatable reason; no candidate passes silently. A reviewer disputing any of Cases 2–4 is forced to dispute a named, typed object — an admission rule, a bridge in R, or a scheme's critical-question inventory — rather than an opaque verdict. That specificity of disagreement is the audit property §4 claims and the falsifiability surface §7 enumerates.

A.7 Concrete Worked Example: ECOA Adverse-Action Notice

The abstract trace in §§A.1–A.6 is sufficient to exhibit the kernel's discipline; this section grounds the same machinery in a real US regulatory regime so the architectural claim is not left abstract. The example is the

adverse-action notice obligation under the Equal Credit Opportunity Act and its implementing regulation, Regulation B (12 CFR §1002.9). When a creditor takes adverse action on a credit application, the creditor must provide the applicant with a statement of the *specific reasons* for the adverse action, drawn from the actual factors that caused the decision. Reciting generic risk language, citing factors that were not actually load-bearing, or omitting a factor that was load-bearing each constitute distinct compliance failures with distinct remediation paths.

The scenario is a small-business credit application denied by an automated underwriting system. A regulated-output artifact — the adverse-action notice — must be emitted. The Catalog Registry R contains, among others, one bridge:

- $b_{\text{aa}} : \sigma_{\text{aa}}$, instantiating an *adverse-action-reason* scheme whose declared signature is

$$\sigma_{\text{aa}}(\mathit{factor}, \mathit{decision}, \mathit{causation}) \vdash \mathit{disclosed\_reason}.$$

  The scheme declares three critical questions: cq_{factual} (*is the cited factor a fact admitted into F from the applicant's file?*), cq_{causal} (*did the cited factor in fact contribute to the adverse decision under the model's documented decision logic?*), and cq_{specific} (*is the cited factor specific in the sense Regulation B requires, rather than a generic risk category?*).

Compile case. The applicant's file admits facts including p_{dscr} (debt-service-coverage ratio below threshold, with the threshold itself an admitted fact p_{thresh} from the lender's documented underwriting policy), p_{dec} (the adverse decision was made), and p_{contrib} (the model's decision-logic record shows p_{dscr} as a top-three contributing factor). The candidate inference is

$$c_{\text{aa}} = (b_{\text{aa}};\ p_{\text{dscr}}, p_{\text{dec}}, p_{\text{contrib}};\ \mathit{disclosed\_reason}),$$

where *disclosed_reason* = "debt-service-coverage ratio below underwriting threshold." All three critical questions discharge against admitted facts: cq_{factual} against p_{dscr}, cq_{causal} against p_{contrib}, cq_{specific} against the conjunction of p_{dscr} and p_{thresh} (a numeric threshold against a numeric ratio is specific in the Regulation B sense, where "insufficient income" alone would not be). The support DAG is acyclic. $K(c_{\text{aa}}) = a_{\text{aa}}$, signed and emitted as a regulated-output artifact suitable for inclusion in the adverse-action notice.

Refusal $\perp_{\textit{missing-fact}}$. The proposer attempts a candidate citing "insufficient credit history" as a reason, but no fact $p_{\text{cred\_hist}}$ has been admitted into F from the applicant's file (the file contains credit-bureau data showing established history; the proposer is hallucinating the factor). $K = \perp_{\text{missing-fact}}$, with the missing leaf named. The compliance significance: a Regulation B violation of citing a non-factual reason is structurally prevented at the Fact Register layer rather than caught downstream by a fair-lending audit.

Refusal $\perp_{\textit{unanchored-bridge}}$. The proposer attempts a candidate citing p_{dscr} as the reason via a substitute bridge b'_{aa} that declares the cited factor as causally contributing without anchoring through the model's decision-logic record. p_{dscr} admits; b'_{aa} requires a *contribution_evidence* leaf that the proposer attempts to satisfy by inference from the existence of the adverse decision rather than by routing through admitted decision-logic facts. $K = \perp_{\text{unanchored-bridge}}$, with the unanchored leaf and the responsible bridge both named. The compliance significance: a *post-hoc rationalization* of an adverse-action reason — citing a factor that exists in the file but was not actually load-bearing in the decision — is structurally prevented at the bridge–register interface, the failure mode that fair-lending litigation has historically struggled to detect.

Refusal $\perp_{\textit{unmet-critical-question}}$. The proposer attempts a candidate citing "insufficient income" as the reason. The fact p_{income} admits; the bridge b_{aa} types correctly; cq_{factual} and cq_{causal} discharge. cq_{specific} does not

discharge: "insufficient income" without a specific income threshold or income-to-obligation ratio is the generic-category failure Regulation B specifically prohibits. The proposer attempts to mark cq_{specific} as carried forward under a Carneades burden allocation; this would be admissible at an intermediate-graph boundary but is refused at the regulated-output boundary (§1.2, §4). $K = \perp_{\text{unmet-critical-question}}$, with cq_{specific} named. The compliance significance: the *specificity* requirement of Regulation B is enforced at the argumentation layer, where the cognitive type of the disclosed reason — specific factor versus generic category — is the structurally checkable property.

What this example demonstrates. The three failure modes §§A.3–A.5 specified abstractly correspond, in the ECOA setting, to three concrete and historically documented compliance failures: hallucinated reasons, post-hoc rationalization, and generic-category disclosures. Each is refused at a structurally distinct layer of the architecture; each refusal localizes to a named, typed object an auditor can find. A regulator reading the kernel's refusal log is not reading a confidence score or a model-uncertainty estimate; the regulator is reading an enumeration of which Regulation B requirement was not satisfied, by which candidate, against which admitted fact or bridge. The architectural property §1 claims for inspection-paradigm regimes — that quality cannot be inspected into a probabilistic guess — has, in this example, a corresponding positive claim: the ECOA-compliance property *can* be constructed into the inference at production time, by refusing every adverse-action reason that does not type-check against the regulation's structural requirements.