Toward Cognitive State Machines

Exploring Foundations and Formal Concepts of Machine Reasoning

Dr.-Ing. Christian Gilcher, Advanced Cognitive Systems Lab, Karlsruhe
Technical Research Reflections #002, 30. October 2025

Scientific Disclaimer: This document is an agenda-setting outlook on the future ACSL research program on Cognitive Architectures. It summarizes the current stage of our work toward formal, verifiable models of machine reasoning. The text should be read as an open hypothesis rather than a finalized theory.

Abstract

This paper explores and introduces the idea of a Cognitive State Machine (CSM) -- a working hypothesis that asks whether machine reasoning can be described as a structured, verifiable process. By now it has become clear that the limits of today's generative systems are not computational -- they are structural.

Over the past three years, that realization has matured from an operational insight into a guiding architectural principle.

What began as an observation in practice has since become a question of architecture. The attempt to engineer reliable reasoning systems forced us into seeking formalization. In that sense, theoretical reflection became unavoidable, not intentional.

The Cognitive State Machine (CSM) therefore represents a hypothesis discovered through engineering -- not engineering derived from theory. In walking this path, we found ourselves building not only software, but a theoretical model. What began as an engineering effort became the search for a formal framework of Machine Reasoning.

The paper presents initial thoughts on conceptual CSM foundations, the connect to its architectural instantiation through the Leibniz–von Neumann Architecture (LVNA), and empirical results from early-stage production systems (e1, e2) operating under regulatory conditions. We treat CSM as an engineering hypothesis, subject to formalization and falsification.

What began as engineering has become theory, and what now follows is the convergence of both: building systems to formalize what theory reveals, and theorizing what construction makes visible. Our objective is not to redefine intelligence, but to test how far reasoning can be expressed as a verifiable computational structure.

In retrospect, the field's fascination with scale now appears as a transitional phase -- necessary to expose the deeper need for structure. If our hypothesis substantiates, it could form the basis for a new architectural layer of Machine Reasoning -- one defined not by scale, but by structure. It took many quarters of building before this distinction became visible – and now that we see it, our future direction becomes even more obvious.

1. INTRODUCTION: THE AI ADOPTION GAP

In 2022, we began building AI for deployment in regulated industries -- energy, finance, the public sector. These contexts share a non-negotiable requirement: every decision must be auditable, every line of reasoning reproducible, every failure observable.

Especially in Europe, these requirements are non-negotiable necessities. Building intelligent systems within such constraints was never a luxury project, but a response to real regulatory and societal expectations. Over time, we learned that these constraints were not obstacles to innovation -- they were the conditions that shaped it. In other words: what might appear ambitious from afar simply emerged from the environment we work in.

Meeting those standards forced us deeper than expected. The limitation was not intelligence per se -- it was the absence of structure and system architecture.

At first, we considered this a purely architectural gap. But as we tried to fix it, the gap turned into a conceptual one. The architecture forced us to make explicit what had so far remained implicit -- the structure of reasoning itself. At that point, we did not imagine we were formulating a new theoretical path -- we were simply trying to make our systems work. It was not a move from engineering to theory, but the realization that engineering had already become theory in disguise.

We realized: Large language models can predict, generalize, improvise -- but they cannot prove what they know. Token prediction is a continuous statistical process: it has no concepts or structural elements for discrete states, no explicit transitions, no structural validation.

And that means something simple but profound: **You cannot trace a probability distribution. You cannot certify an approximation. You cannot bound the risk of an opaque process.** This was the starting point of what we now call cognitive control -- the ability of a system to reason within constraints that are themselves formally defined.

Instead of accepting that gap, we started building — not another model, but the missing architecture. Long before the term entered the discourse, we were already building systems that embodied it. Step by step, a new foundation emerged:

- The **Cognitive State Machine**, a formal model of structured reasoning built on discrete states, operator-mediated transitions, and formal control;
- The **Leibniz-von-Neumann Architecture**, which translates those principles into a practical system design;
- And our production systems, **e1** and **e2**, which prove the model under real-world conditions -- at scale, in regulation-bound industries.

Together, they form what we call a **Cognitive Architecture** -- an integrated framework for systems that can *reason* in verifiable ways. Not bigger models. Not better prompts. A different class of system.

To our knowledge, no other framework combines theory, architecture, and production at this level of integration. Europe's constraints forced us to go deeper -- and by doing so, they may have opened a path others not yet had to explore.

The following pages summarize what we've learned so far, where the work stands today, and why we believe this marks the beginning of a new architectural era for AI.

Whether this becomes a standard or remains a singular path is not for us to decide – as of today, we can only share our learnings and early practical *and* theoretical evidence from being three years into that space.

2. WHAT LARGE-SCALE AI ADOPTION ACTUALLY REQUIRES

Before describing our approach, we must be precise about what high-stakes deployment actually demands. These requirements emerge from three years of working with enterprises in regulated industries, from regulatory frameworks in energy, finance, and healthcare, and from production experience where failures have real consequences.

These questions reflect four fundamental requirements that current AI approaches cannot provide, structurally.

Traceability means that every reasoning step must be reconstructible. Not approximately, not statistically, but exactly. If a system recommends a specific action, an auditor must be able to trace backward through the complete chain of reasoning that led to that recommendation. This is not about explainability in the sense of post-hoc rationalization. It is about intrinsic structure that preserves the path from input to output.

Validation means that every transformation in the reasoning process must be verified against explicit rules or constraints. When a system moves from one reasoning state to another, that transition must be checked. This verification must happen during the reasoning process, not after. A system cannot be validated by examining only its outputs. It must be validated by examining its structure.

Reproducibility means that the same input must produce the same reasoning structure. Not necessarily the same final answer, but the same path through the reasoning space. This enables testing, certification, and insurance. A system whose behavior is fundamentally non-deterministic cannot be certified for high-stakes use, because certification applies to specific behaviors, not to probability distributions over behaviors.

Observable failures mean that when a system encounters a situation it cannot handle, it must recognize this and escalate rather than proceed. Current AI systems have no reliable mechanism for distinguishing between situations they can handle and situations they cannot. They produce output with equal confidence regardless of

whether they are operating within their competence boundary or far outside it. For deployment in contexts where errors have serious consequences, this is unacceptable.

These four properties are not independent. They form a coherent structural requirement. Traceability requires discrete states. Validation requires explicit transitions. Reproducibility requires deterministic structure. Observable failures require introspective control that operates on explicit state.

Token prediction, in its current architectural form, cannot provide these properties. This is not a criticism of large language models per se. It is a statement about the architectural structure of the approach. A system that generates output by predicting the next most likely token, repeatedly, has no discrete states to trace, no explicit transitions to validate, no deterministic structure to reproduce, and no introspective awareness of its competence boundaries.

We encountered this gap not as a theoretical problem but as a practical barrier. Every client conversation ended the same way. Impressive demonstrations, followed by questions about auditability and reproducibility, followed by silence. The gap between capability and deployment is not closing through better prompts or larger models. It is structural.

The question became: what kind of system architecture can provide these properties while still leveraging the remarkable capabilities of large language models? The answer required going deeper than we initially expected.

3. COGNITIVE STATE MACHINES: THEORETICAL FOUNDATIONS

When we started, we initially did not set out to develop a new theoretical model. We simply set out to build systems that could be deployed in regulated industries. But the domain forced us deeper. To provide traceability, validation, reproducibility, and observable failures, we needed discrete states, explicit transitions, and formal control. As we built systems with these properties, we realized we were implementing something that had no formal definition.

Cognitive State Machines emerged from this work. CSM is our approach to formalize our discovery: a computational model for structured reasoning that could become as fundamental to cognitive systems as Turing Machines are to computing systems.

A Cognitive State Machine operates on three core primitives:

- Cognitive Artefacts,
- Cognitive Operators, and
- Cognitive Control.

Formally, a Cognitive State Machine can be defined as

$$CSM = (A, O, C, S)$$

where
A denotes the set of artefacts,
O the set of cognitive operators,
C the control function, and
$S_t \subseteq A$ the cognitive state of the system at time t.

Cognitive Artefacts are discrete, versioned units of thoughts (represented through a sequence of tokens). However, they are treated as discrete objects with explicit structure. Each artefact has content, provenance, status, and timestamp. An artefact might represent a claim, a piece of evidence, an inference, a constraint, or a decision. The critical property is discreteness. The artefact is a first-class computational entity with defined boundaries.

This discreteness is fundamental. In token prediction systems, there are no structural boundaries. The system generates a continuous stream of tokens, and any segmentation into reasoning units is imposed through interpretation after the fact. In CSM, boundaries, state and transitions are part of the formal structure.

Cognitive Operators are functions that transform artefacts. An operator takes one or more artefacts as input and produces one or more artefacts as output. Each operator O_j defines a transformation

$$O_j: A_i \rightarrow A_k$$

or, for composite reasoning steps,

$$O_j: A_i \times A_m \rightarrow A_k$$

The operator's execution yields a validation signature

$$\sigma_j = f(O_j, A_i, A_k, t)$$

capturing provenance and timestamp for traceability. Critically, this transformation is not arbitrary. Each operator has a defined type and produces a validation signature.

In current implementations, these operators are expressed in natural language but executed semantically by the underlying language model. The model performs the transformation defined by the operator's rule text, and the resulting state transition is validated by the surrounding control logic. This ensures that linguistic expressiveness and formal control coexist within the same reasoning loop.

Each operator transformation generates provenance. When operator O transforms artefact A into artefact B, the system records this transformation with a cryptographic signature. This creates an immutable chain: B was produced by O applied to A at timestamp T. This provenance chain is the foundation of traceability.

Cognitive Control orchestrates operators and enforces constraints. It determines which operators to apply, in what sequence, under what conditions. Cognitive Control can be expressed as a normative function

$$C: (A_i, O_j, A_k) \rightarrow \{accept, reject, revise\}$$

selecting valid transitions under explicit constraints. It checks preconditions before operator application and validates postconditions after. It detects when the system has reached a state where no valid operator can proceed, and it escalates.

Control is normative, not statistical. It does not select the next operator based on what is most likely to produce good output. It selects based on what is valid given the current state and the system's constraints. This is a fundamental difference from how current AI systems work.

Once finished, we can use CSM to ask questions like: for any given problem *X*, within a conceptual space *Y* and a set of cognitive operators *Z*, what are the conditions under which *X* is computationally solvable.

Formally, for a problem X, within conceptual domain Y and operator set Z,

$$Solvable(X) \iff Coverage(Y) \wedge Completeness(Z)$$

or, equivalently,

$$\exists (Y, Z): X \text{ is solvable within } CSM(Y, Z)$$

We have started to work on establishing a cognitive analogy to Turing completeness, where solvability would depend on conceptual coverage and operator sufficiency. In other words: if this hypothesis holds true, reasoning can become a question of *coverage*: how far does *Y* span the relevant conceptual domain, and to what extent does the composition of *Z* provide sufficient transformations over *Y* to resolve *X*.

But as of today, CSM is not yet fully formalized. We are working on that rigorously through the Advanced Cognitive Systems Lab. But the core principles are clear and have been validated through implementation and deployment.

The relationship between the three core primitives defines CSM: a set of artefacts, applies operators to transform them, and uses control to orchestrate these transformations under explicit constraints. At any point in time, the system is in a specific cognitive state, defined by its current set of artefacts. The system transitions between states by applying operators. Each transition is discrete, explicit, and validated.

CSM is <u>not</u> a replacement for large language models. LLMs are extraordinarily powerful at pattern recognition, language generation, and approximate reasoning. CSM provides a formal structure within which these capabilities can be deployed while maintaining the structural properties that high-stakes contexts demand.

In CSM-based systems, an LLM might generate candidate artefacts, suggest operator applications, or evaluate the quality of transformations. But the LLM operates as a component within the CSM structure. The artefacts, operators, and control are not generated by token prediction. They are formal elements of the system architecture.

This transition -- from building systems to formalizing their underlying structure -- defines the methodological stance of this research: theory follows construction, and construction tests theory.

CSM is presented as a hypothesis -- not detached from engineering, but evolving through it. Its purpose is to test, in practice, how far structured reasoning can be captured within a finite, verifiable state system -- and to refine the theory by building what it describes.

4. LEIBNIZ-VON-NEUMANN ARCHITECTURE: THE BLUEPRINT

CSM is an attempt to provide a theoretical frame for cognitive systems. Building them requires an architectural blueprint. The Leibniz-von-Neumann Architecture (LVNA) is that blueprint. It derived from our practical work in real-life customer projects.

The reference to *Leibniz* in the Leibniz-von-Neumann Architecture is intentional: it reflects the idea that reasoning, like computation, can be expressed through formal operations on discrete symbols -- a concept Leibniz anticipated centuries before digital machines.

LVNA emerged from the same practical work that led to CSM. As we built systems for regulated industries, we discovered architectural patterns that worked. Event-driven operation. Persistent state storage. Explicit operator services. Control mechanisms that orchestrate and validate. These patterns proved themselves in production before we recognized them as an architectural framework.

In August 2025, we published an initial paper on LVNA. Here we present only the core principles and their relationship to CSM (although the name „CSM“ didn’t exist back then).

LVNA is built on three architectural decisions: event-driven operation, state persistence, and operator-based transformation. These decisions correspond with CSM requirements and distinguish LVNA from conventional AI architectures.

Event-driven means that the system operates by processing discrete events rather than by continuous execution. Each operator application is an event. Each artefact creation is an event. Each state transition is an event. This maps directly to the discrete nature of CSM states and transitions. The event stream is the system's operational foundation.

State-persistent means that cognitive artefacts are stored durably, not held ephemerally in memory. LVNA uses Apache Kafka as both event bus and persistent memory. Every artefact, every transformation, every provenance chain is written to Kafka

topics and persists. This enables complete reconstruction of any reasoning path. It also enables the system to be stopped, inspected, and resumed without loss of cognitive state. While technology like Kafka is, per se, an industry standard, its role in LVNA is epistemic -- preserving the continuity of reasoning itself.

Operator-based means that all reasoning transformations are mediated by explicit operators. There is no unstructured generation of cognitive content. When the system needs to create a new artefact or transform an existing one, it invokes a defined operator. Operators are implemented as services that receive artefacts as input and produce artefacts as output, with full provenance tracking.

The **Cognitive Control Unit** is the orchestration layer. It monitors the current cognitive state, determines which operators can validly be applied, selects operators based on control policies, and validates that operator outputs satisfy constraints. The CCU does not generate cognitive content. It enforces cognitive integrity.

This separation is fundamental. In conventional AI systems, the same mechanism that generates content also determines what content to generate. In LVNA, these are separated. Operators generate content. Control determines which operators to invoke and validates their outputs. This separation enables the structural properties that CSM requires.

LVNA is software-defined. It runs on standard cloud infrastructure using commodity components. This is deliberate. Cognitive systems should be as portable and scalable as computing systems.

The architecture accommodates large language models as operator implementations. An operator might use an LLM to generate candidate artefacts, to evaluate the relevance of evidence, or to suggest analogies. But the LLM operates within the operator interface. It receives artefacts as input and produces artefacts as output. The control layer validates these outputs before they become part of the cognitive state.

This is how LVNA bridges the gap between token prediction and structured reasoning. It does not reject the capabilities of LLMs. It provides a structure within which those capabilities can be used while maintaining cognitive integrity.

These LVNA components -- artefact management, operator orchestration, control services, provenance tracking, validation -- constitute what we call **Cognitive Infrastructure**.

The sum of all LVNA components, we call ‚**Cognitive Infrastructure**‘, coining a new technology category: it is, in reality, the deployment layer that makes cognitive systems production-ready. Not better models, but infrastructure for production-grade AI.

5. PRODUCTION VALIDATION: E1 AND E2

The first real-life implementations of LVNA are e1 and e2. These cognitive systems have been running in production since 2024, handling real workloads in regulated industries where failures have consequences.

e1 operates in energy contexts at RWE, one of Europe's largest energy companies. The system processes real-time market data and provides decision support for trading and grid optimization under regulatory constraints. Every recommendation must be auditable. Every decision path must be reconstructible.

e2 operates in industrial automation (Rwe, MVV Netze), financial services (Hauck Aufhäuser Lampe, BWGV) and Public Sector (within non-disclosed organisations). In all contexts, the requirements are similar: structured reasoning under explicit constraints, complete traceability, reproducible behavior, and the ability to recognize and escalate situations the system cannot handle.

e1 and e2 are research prototypes in terms of using them for continued experimentations on LVNA and CSM, but they are robust, monitored and controlled production systems. They already solve immediate problems for clients.

We identified and measured three critical properties over multiple years of operation.

Reproducibility was measured by providing the same inputs multiple times and examining whether the system produced the same cognitive state sequences. Across thousands of test cases, the systems demonstrated 95 percent reasoning reproducibility, measured across hundreds of dedicated test cases at the customers‘ domain. The same input produces the same sequence of cognitive artefacts, the same operator applications, the same final state.

The five percent non-reproducibility is not random variation. It occurs in specific, identifiable situations: when the system deliberately introduces bounded randomness for exploration, when human operators inject new constraints during processing, or when external data sources return different values at different times. These are designed behaviors, not failures of determinism. The system's structure is reproducible. Its interaction with non-deterministic external inputs is explicitly controlled.

Causality tracing was measured by randomly selecting decisions and reconstructing their complete reasoning paths. Every operator application is logged with full provenance. Every artefact records its creation operator, input artefacts, and timestamp. Auditors can trace backward from any decision to the complete chain of reasoning that produced it. Across all production deployments, one hundred percent of decisions have been successfully traced. There are no gaps in the provenance chain.

Bounded risk was assessed by measuring the system's behavior on inputs outside its training distribution. In conventional AI systems, out-of-distribution inputs are the primary source of unquantified risk. The system may produce confident but incorrect outputs with no indication that it has exceeded its competence boundary.

In e1 and e2, out-of-distribution inputs trigger observable failures. The control layer detects that no valid operator can proceed, or that operator outputs violate constraints, and the system escalates. It does not attempt to generate plausible-looking output. It explicitly acknowledges that it has reached the boundary of its defined competence.

This enables bounded risk quantification. An enterprise deploying these systems can define the input space, measure the system's behavior across that space, and identify the boundaries where failures become observable. Risk is not eliminated, but will be bounded and quantifiable. This can possibly bring us toward insurance coverage for these deployments, something that is not currently possible for conventional AI systems in comparable high-stakes contexts.

The production experience has also revealed limitations and costs. Cognitive systems like e1 and e2 are more complex to build than conventional AI systems. They require explicit operator design, constraint specification, and control policy definition. This is more labor-intensive than training a model end-to-end. The trade-off is between upfront design effort and deployment risk.

Cognitive systems are also slower. Maintaining explicit state, validating every transition, and checking constraints adds computational overhead. For applications where response time is critical and deployment risk is low, this overhead may not be justified. For applications where correctness and auditability are mandatory, it is a necessary cost.

To our knowledge, e1 and e2 are the first production cognitive systems operating in regulated industries within these properties named above. We did not build them to be first. We built them because clients needed systems that could actually be deployed. But the result is that we have operational experience that does not exist elsewhere.

Each deployment not only validates the architecture, but extends the theory. Every trace, every failure, every reproducible reasoning path refines our understanding of what CSM must capture. The production systems are not end points -- they are the experimental apparatus of the theory itself.

6. IMPLICATIONS

Cognitive systems and the infrastructure that supports them have implications across multiple domains. We examine four: research, model development, industry adoption, and regulation.

For **research**, CSM opens questions that need rigorous work. The model we have defined is based on production experience, not formal derivation. Formalizing it properly requires answering questions we have not yet fully addressed. What is the minimal set of operator types required for general cognitive computation? How do we formally verify that a set of constraints is satisfiable? What are the complexity bounds for cognitive state reachability? How do we compose operators while preserving validation properties?

These questions are analogous to the questions that followed early computational models: decidability, complexity classes, automata theory. CSM provides a starting point, but the theory requires substantial development. This work will benefit from collaboration across computer science, logic, and cognitive science. We are pursuing this through ACSL, but we recognize it is larger than any single organization.

Research into cognitive systems also benefits AI model development. When models operate as components within cognitive infrastructure, their requirements become clearer. A model that generates candidate artefacts needs different capabilities than a model that validates transformations. A model that operates under explicit constraints needs different characteristics than a model that optimizes for output quality alone.

For **model developers and vendors**, cognitive infrastructure provides opportunity rather than competition. As enterprises adopt cognitive infrastructure for high-stakes deployment, demand for capable models increases. Models become more valuable when they can be deployed safely, and cognitive infrastructure enables that deployment.

This creates pressure and opportunity in model development. Models that can expose their uncertainty, that can operate under constraints, that can indicate when they are outside their training distribution become more valuable in cognitive infrastructure contexts. Market demand will drive capability development in these directions.

For **industry**, cognitive infrastructure provides a practical adoption path for AI in high-stakes contexts. The current situation is untenable. Enterprises see AI capabilities but cannot deploy them where the stakes are highest and the regulatory requirements are strictest. Cognitive infrastructure addresses this by providing the structural properties that deployment requires.

Adoption will likely follow the pattern of previous infrastructure transitions. Early adopters in regulated industries validate the approach. Tooling and standardization mature. Costs decrease through economies of scale and competition. More industries adopt. Eventually, cognitive infrastructure becomes the default approach for enterprise AI deployment in any context where governance and risk management matter.

For **regulation**, cognitive infrastructure enables different approaches to AI governance. Current regulatory frameworks struggle with AI because the systems are opaque and non-deterministic. Regulators can require documentation and testing, but they cannot verify that a system actually behaves as documented.

Cognitive systems could fundamentally change this. When every reasoning step is traceable, every transformation is validated, and every constraint is enforced structurally, compliance becomes verifiable. A regulator can examine the cognitive state, inspect the operators, and validate that constraints are enforced during operation, not just claimed in documentation.

Regulatory pressure can drive innovation rather than hinder it. The European Union's AI Act and similar frameworks worldwide create strong incentives for structural approaches to AI safety and compliance. Cognitive infrastructure emerged partly from these regulatory requirements. Organizations building systems for regulated deployment must provide auditability, transparency, and risk management. Cognitive infrastructure provides these properties structurally.

Further work will extend Cognitive Control with explicit epistemic grounding -- anchoring reasoning not only in logical structure, but in structured knowledge itself.

From an engineering standpoint, this work proceeds empirically: by building, observing, and measuring. We do not claim that reasoning *is* a state machine; rather, we are discovering how far reasoning can be engineered as one. The distinction is subtle but crucial -- we build to learn, and we formalize to make learning durable.

7. CONCLUSION AND OUTLOOK

We didn't set out to invent a new class of AI systems. We simply needed systems that could be trusted — and built the architecture they required.

Cognitive State Machines define what structured, verifiable reasoning means in computational terms: discrete states, operator-mediated transitions, and formal control.

The **Leibniz-von-Neumann Architecture** translates those principles into real systems that run on standard infrastructure.

e1 and **e2** have proven that this architecture works -- not in theory, but in production, under regulation, where failure has consequences.

Together, they form a **Cognitive Architecture**: an integrated framework that bridges the gap between AI capability and real-world adoption. Not by making models bigger, but by giving them structure -- the kind that can be traced, validated, and certified.

The work continues. CSM is still being formalized. LVNA keeps evolving through deployment. The cognitive infrastructure beneath both is expanding toward standardization. But the foundations are solid.

The deployment gap between AI capability and trust will not close through scale -- it will close through architecture. That is the path we are on. Perhaps that is the most European part of this story -- progress not through scale, but through system structure and architecture.

Advanced Cognitive Systems is not an experiment in AI. In retrospect, the implications reach beyond our initial intent -- it could be the beginning of a new architecture layer for intelligence itself.

Where Turing formalized computation and von Neumann laid the foundation for its industrialization, Cognitive State Machines adapt that analogy to reasoning itself.

We began as engineers solving immediate deployment problems. But solving them put us on a path towards developing a theory -- one that did not yet exist. The two lines become symbiotic: Practice is being abstracted into theory; theory stabilizes practice.

We do not ask for applause or approval. We share our motivation, our rationale and the direction of our work. In retrospect, the field's fascination with scale now appears as a transitional phase -- necessary to expose the deeper need for structure. And although we have already gone deep, it still feels like we are just getting started.

If this path holds, it will not just define how machines reason -- it will redefine our understanding of cognition itself.

Recommended Readings

1. Turing, A. M. (1936).
On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42(2), 230–265.

2. Von Neumann, J. (1945).
First Draft of a Report on the EDVAC. IEEE Annals of the History of Computing, 15(4), 27–75.

3. Robinson, J. A. (1965).
A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM, 12(1), 23–41.

4. Kowalski, R. (1979).
Logic for Problem Solving. North-Holland.

5. Bench-Capon, T. J. M., & Sartor, G. (2003).
A Model of Legal Reasoning with Cases Incorporating Theories and Values. Artificial Intelligence, 150(1–2), 97–143.

6. Prakken, H., & Sartor, G. (2015).
Law and Logic: A Review from an Argumentation Perspective. Artificial Intelligence, 227, 214–245.

7. Governatori, G., et al. (2013).
Compliance Checking in Business Process Models. IEEE Transactions on Software Engineering, 39(6), 744–757.

8. Julieth Patricia Castellanos Ardila, et al. (2022).
Compliance Checking of Software Processes: A Systematic Literature Review. Information Systems, 75, 1–20.

9. Garcez, A. S. d., et al. (2019).
Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning. Journal of Applied Logic, 6, 100–109.

10. Manhaeve, R., et al. (2018).
DeepProbLog: Neural Probabilistic Logic Programming. Advances in Neural Information Processing Systems, 31, 3749–3759.

11. Lewis, P., et al. (2020).
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

12. Arora, S., et al. (2022).
Why Exposure Bias Matters: An Imitation Learning Perspective of Error Accumulation in Language Generation. arXiv:2204.01171.

13. Clarke, E. M., et al. (2018).
Model Checking. MIT Press.

14. Forgy, C. L. (1982).
Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Artificial Intelligence, 19(1), 17–37.

15. Chang, C. L., & Lee, R. C. T. (1973).
Symbolic Logic and Mechanical Theorem Proving. Academic Press.

16. Steen, A., & Benzmüller, C. (2024).
Non-Classical Reasoning for Contemporary AI Applications. KI – Künstliche Intelligenz, 38(4), 299–304.

17. Lee, A., & Tong, H. (2025).
Reinforcement Learning for LLM Reasoning Under Memory Constraints. arXiv preprint.

18. Gasser, U. (2024).
"Navigating AI Governance as a Normative Field: Norms, Patterns, and Dynamics." In *Realizing the Promise and Minimizing the Perils of AI for Science and the Scientific Community.* University of Pennsylvania Press.

19. Petrov, M., & Vechev, M. (2025).
Proof or Bluff? Evaluating LLMs on the 2025 USA Math Olympiad. Conference on Mathematical Foundations of Program Semantics.

20. Tran, Q., et al. (2024).
RARE: Retrieval-Augmented Reasoning Enhancement for Large Language Models. arXiv:2412.02830.

About ACSL

Advanced Cognitive Systems Lab is a privately funded, non-for-profit research lab based in Karlsruhe, Germany. Founded in July 2025 as a spin-off from embraceable Technology, a pioneer in conceptualizing and crafting cognitive systems, ACSL puts its focus on foundational aspects on Machine Reasoning and Machine Cognition in general.

(End of document)