Representation Models (RM): Explicit Representation, Controlled Reasoning, and beyond Language-Centric AI

Simone Mazzoni
ORCID: 0009-0004-0400-4823

January 2026 (rev. v1.2)

Abstract

Recent advances in Artificial Intelligence have been driven by large-scale statistical models operating over implicit representational spaces, with Large Language Models (LLMs) as the most prominent example. While such systems demonstrate impressive capabilities across a range of tasks, their reliance on implicit representations and probabilistic continuation limits transparency, auditability, and control in safety-critical or regulated domains.

This paper introduces *Representation Models* (RMs), a new class of AI systems in which explicit internal representations constitute the primary computational substrate. In an RM, reasoning, learning, and decision-making proceed through the construction, transformation, and stabilization of representations under declared constraints, rather than through modality-specific prediction alone. RMs separate validated knowledge from exploratory computation, enforce structural invariants, and organize computation across hierarchical representational orders.

We provide a precise architectural definition of RMs, clarify the distinction between knowledge, learning, reasoning, and evolution, and position RMs relative to language models, world models, and symbolic systems. This document serves as a foundational public introduction of the RM paradigm, focusing on architectural principles rather than performance benchmarks, and establishing the conceptual framework required for subsequent instantiations and validation phases.

1 Introduction

The last decade of Artificial Intelligence has been dominated by the rapid rise of large-scale statistical models operating over implicit representational spaces, with Large Language Models (LLMs) as the most visible and widely deployed instance. Transformer-based systems trained on massive corpora have achieved unprecedented fluency, versatility, and apparent reasoning ability across a wide range of tasks. These advances have led to the widespread deployment of LLMs in domains such as software development, decision support, education, and content generation.

At the same time, the limitations of language-centric AI have become increasingly visible. LLMs remain prone to hallucinations, logical inconsistency, threshold instability, and brittle behavior under adversarial or out-of-distribution inputs. These issues persist despite continued scaling of model size, data volume, and computational resources. Empirically, performance gains exhibit diminishing returns, while operational costs grow rapidly.

These limitations are not incidental. They arise from the core architectural choice underlying LLMs: the treatment of language as both the input modality and the internal substrate of reasoning. In LLMs, all cognitive activity—semantic interpretation, inference, planning, and explanation—is

projected into a single, flat representational space optimized for probabilistic token prediction. This architectural flattening obscures validity conditions, hides thresholds, and prevents explicit enforcement of invariants.

This paper argues that further progress requires a structural shift at the level of computation itself. Rather than attempting to extract ever more competence from implicit, modality-bound representations, we introduce *Representation Models* (RMs): AI systems in which explicit, structured, and hierarchical internal representations form the primary substrate of computation.

Representation Models (RM). In this work, we introduce the notion of a *Representation Model (RM)* to denote a general class of artificial systems in which *explicit internal representations constitute the primary computational substrate.* In an RM, reasoning, learning, and decision-making proceed through the construction, transformation, and stabilization of representations under intrinsic constraints, rather than through implicit statistical continuation over untyped state spaces. The architectural principles developed in this paper define such Representation Models in the context of artificial intelligence.

2 Representation-Centric Intelligence

2.1 From Language-Centric to Representation-Centric AI

In LLM-based systems, intelligence is implicitly equated with the ability to generate linguistically plausible continuations. The internal state of the model is a high-dimensional embedding space whose geometry is shaped by statistical regularities in training data. While this space encodes rich information, its structure is implicit: neither the semantics of its dimensions nor the validity of its inferences are directly accessible or controllable.

By contrast, a representation-centric AI system treats internal representations as first-class computational objects. Language becomes one representational modality among others, rather than the universal medium of cognition. Reasoning proceeds by constructing, transforming, and validating explicit representational structures that encode facts, constraints, relations, and invariants.

Representation Models are defined by this shift in focus. An RM does not ask, "What is the most probable continuation of this text?" but rather, "What representation of the problem domain is consistent, stable, and valid under the applicable constraints?"

Computational Space Instantiations. Within the class of Representation Models, different domains correspond to distinct *computational space instantiations*, characterized by their representational primitives, admissible operations, and closure conditions. For example, *Language Representation Models (LRMs)* correspond to RMs specialized for linguistic structures; *Mathematical Representation Models* instantiate formal computational spaces governed by consistency, invariance, and derivability; and *World Representation Models* instantiate continuous computational spaces governed by physical causality and dynamics. These instantiations differ not by learning mechanism, but by the structure of their representational spaces and the criteria under which representational closure is achieved.

2.2 Hierarchical Decomposition and Computation Orders

A central feature of RMs is hierarchical decomposition. Problems are not solved in a single undifferentiated space, but are decomposed across *computation Orders* (or simply *Orders*), each defining a coherent representational and computational regime.

This ordered organization prevents category errors, such as resolving normative conflicts with statistical heuristics or treating vague predicates as crisp logical facts. It also provides a natural framework for scaling competence: complexity is absorbed by vertical abstraction rather than horizontal parameter expansion.

3 Knowledge, Learning, Reasoning, and Evolution

To clarify the scope of RMs, it is essential to distinguish four fundamental modes of computational activity: knowledge, learning, reasoning, and evolution. These terms are often conflated in discussions of AI, leading to conceptual confusion and inflated claims.

3.1 Knowledge

Knowledge refers to the set of validated, stationary representations available to a system at a given moment. Knowledge consists of closed structures: facts, rules, invariants, and constraints that have passed validation and are treated as stable within the current representational order.

In RMs, knowledge is explicitly represented and stored. It is not merely an emergent property of model weights, but a structured substrate that can be inspected, audited, and constrained.

3.2 Learning

Learning is the process by which a system incorporates validated structures into its existing representational regime. Learning aligns new information with existing representations but does not, by itself, create new representational dimensions.

Both LLMs and RMs learn. In LLMs, learning occurs through gradient-based updates that shape an implicit representational space. In RMs, learning may include the validation and integration of new axioms, rules, or thresholds into an explicit substrate.

3.3 Reasoning

Reasoning is linear computation performed within an already stabilized representational space. It consists of deriving consequences, resolving configurations, and evaluating propositions under existing closure constraints.

Under this definition, LLMs do reason: they perform complex computations within a learned representational space. However, this representational space is implicit, opaque, and not directly governable. As a result, LLM reasoning cannot reliably enforce domain-specific invariants, expose validity thresholds, or guarantee closure.

RMs also reason, but they do so over explicit representations. This makes reasoning traceable, auditable, and controllable.

3.4 Evolution

Evolution occurs when residual non-closure cannot be resolved by learning or reasoning within the current representational order. In this case, the system must generate a new representational dimension—a higher Order—in which the previously unresolved structure becomes expressible and solvable.

Standard LLMs do not undergo representational evolution at inference time. Their representational capacity is fixed post-training. RMs, by contrast, are designed to detect non-closure explicitly and, in advanced instantiations, to support controlled representational expansion.

4 Implicit and Explicit Representation: LLMs vs RMs

The fundamental distinction between LLMs and RMs is not the presence or absence of reasoning, but the nature of the representational system over which reasoning is performed.

LLMs reason over an *implicit* representational space induced by large-scale statistical learning. Validity conditions, thresholds, and invariants are distributed across parameters and cannot be isolated or controlled. When an LLM produces an answer, it cannot reliably explain under which assumptions the answer holds, how sensitive it is to threshold variation, or whether alternative constraints would invalidate it.

RMs make the representational system *explicit*. They separate validated knowledge from exploratory computation, represent thresholds and constraints directly, and evaluate propositions against declared invariants. This explicitness enables auditability, governance, and integration with regulatory or domain-specific requirements.

The competence gap between LLMs and RMs is therefore not a gap in computational power, but a gap in representational control. RMs address this gap by design.

5 Hierarchical Representation and Competence Stabilization

5.1 Implicit Flat Representation in LLMs

Large Language Models operate over a rich but implicit representational space induced by large-scale statistical learning. This space encodes semantic, syntactic, and pragmatic regularities, and supports non-trivial forms of reasoning. However, because this representational system is not explicit, it is treated architecturally as a single, flat space.

Learning, reasoning, and inference are all performed within the same undifferentiated geometry. There is no explicit separation between lower-level competencies (e.g., factual regularities, basic logical relations) and higher-level abstractions (e.g., narrative coherence, rhetorical structure, speculative reasoning). As a consequence, higher-level inference can override or contradict lower-level regularities without triggering any structural violation.

This architectural flatness explains several well-known failure modes of LLMs. In particular, hallucination is not merely the production of false content, but the result of higher-level reasoning collapsing lower-level constraints in the absence of explicit representational protection.

5.2 Explicit Hierarchical Representation in RMs

Representation Models are designed around an explicit and hierarchical representational system. Representations are organized into Orders, each corresponding to a distinct level of competence and abstraction. Within each Order, the system can expand horizontally by enriching its representations, while vertical expansion occurs when higher-order abstractions are required.

Crucially, knowledge acquired at a given Order is stabilized before it can be used by higher Orders. Once validated, lower-level representations become stationary and cannot be invalidated by higher-level reasoning. Higher Orders may reason about, contextualize, or constrain lower-level knowledge, but they cannot overwrite it.

This architecture enables progressive learning. Competence is built incrementally: lower-level knowledge forms the stable foundation upon which higher-level reasoning is constructed. This mirrors the developmental trajectory observed in human cognition, where sensory and perceptual competencies acquired early in life remain stable and are not negated by later abstract reasoning.

5.3 Reasoning Without Competence Collapse

In RMs, reasoning is constrained by the hierarchical structure of representation. When a higher-level inference conflicts with stabilized lower-level knowledge, the system does not emit an incoherent output. Instead, the conflict is detected as a structural non-closure, requiring either refinement within the current Order or escalation to a higher representational Order.

Under this regime, hallucination is structurally prevented. What would appear as a hallucination in a flat architecture is reinterpreted as an invalid inference attempt that violates established competence boundaries.

The fundamental distinction between LLMs and RMs is therefore not the presence of reasoning, but the presence of explicit, hierarchical competence stabilization. RMs ensure that higher-level reasoning reinforces, rather than destabilizes, the knowledge acquired at lower levels.

6 Orders as Computational Spaces and Intrinsic Multi-Agenticity

6.1 Orders as Coherent Computational Spaces

In Representation Models, an *Order* is not merely an abstraction level or a semantic hierarchy. An Order defines a coherent computational space characterized by:

- a specific class of representational objects,
- admissible operations and transformations,
- validity constraints and closure conditions,
- resource bounds appropriate to the problem scale.

Reasoning is always performed within such a computational space. Each Order provides the minimal structure required to resolve a class of problems under its constraints. This design rejects the assumption of a single universal reasoning space and instead treats computation as context-dependent and structurally typed.

6.2 Non-Closure as a First-Class Computational Condition

When reasoning cannot reach closure within the current Order—due to contradiction, ambiguity, or insufficient representational adequacy—the system must not force a resolution.

In Representation Models, non-closure is treated as an explicit computational condition. It indicates that the current representational regime is insufficient *under the active constraints*. The architectural requirement is that the system respond through explicit, governed adaptation rather than implicit completion.

This document does not prescribe any triggering policy, metric, or operational procedure for adaptation. The mechanisms by which a system refines representations, reorganizes its computational regime, or transitions across Orders are implementation-dependent and intentionally outside the scope of this foundational paper.

6.3 Cross-Space Representational Closure

In the architecture introduced in this work, orchestration is not implemented as an external control layer supervising a language model. Instead, it emerges from a *multi-hierarchical Representation*

Model in which multiple representational spaces are jointly embedded within a unified closure framework.

Each representational space defines a specific computational space instantiation, characterized by its representational primitives, admissible operations, and closure conditions. Language models correspond to computational space instantiations specialized for linguistic representation. Other representational spaces may encode logical consistency, mathematical structure, normative constraints, or domain-specific invariants.

Crucially, representational commitments are not finalized within a single space.

We refer to this family of behaviors as *cross-space representational closure*: a regime in which outputs produced within one representational space are treated as provisional configurations and evaluated under the constraints of other representational spaces.

When closure cannot be achieved within the originating space, the resulting non-closure can be represented explicitly across spaces in an observable and governed manner. *This document does not prescribe any operational policy for how cross-space non-closure is handled, nor any triggering mechanism for refinement or escalation.* Such responses are implementation-dependent and intentionally outside the scope of this foundational paper.

6.4 Horizontal and Vertical Expansion

The representational growth of an RM proceeds along two orthogonal axes:

- **Horizontal expansion**: enrichment of representations and knowledge within a given Order, increasing competence without changing the computational space.
- **Vertical expansion**: transition to a higher Order, introducing a computational space with expanded representational capacity.

This classification is conceptual: it does not prescribe how or when expansion is executed, nor any operational policy for triggering or controlling it.

This dual expansion mechanism allows RMs to scale competence efficiently. Complexity is absorbed by structural differentiation rather than brute-force enumeration.

6.5 Intrinsic Multi-Agent Architecture

Because each Order constitutes a coherent computational space with its own constraints and reasoning dynamics, an RM is intrinsically multi-agentic. Each Order functions as a specialized reasoning unit, interacting with other Orders through explicit interfaces.

This multi-agenticity is not implemented by instantiating independent agents or heuristic tool calls. It emerges naturally from the hierarchical organization of computation. Coordination between Orders is governed by representational constraints rather than external orchestration logic.

6.6 Natural Multi-Interface Capability

The hierarchical computational structure of RMs enables simultaneous interaction with multiple interfaces:

- lower Orders can interface with physical sensors, actuators, and signal-processing systems,
- intermediate Orders support cognitive, perceptual, and situational reasoning,
- higher Orders handle symbolic, linguistic, normative, and strategic reasoning.

Because each interface binds to the Order whose computational space matches its representational requirements, integration across physical, cognitive, and symbolic domains is achieved without architectural fragmentation. This positions RMs as a natural foundation for embodied, cognitive, and socio-technical AI systems.

7 Scaling and Non-Closure (Non-Prescriptive)

7.1 On Causes of Non-Closure (Non-Prescriptive)

Non-closure may arise from multiple structural causes, including limits of expressivity, insufficient representational resolution, or conflicting constraints. The purpose of this paper is not to classify such causes operationally, but to require that they be handled explicitly and without forced completion.

Any taxonomy of non-closure causes, and the corresponding operational responses, is implementation-dependent and will be treated in subsequent publications.

8 Knowledge and Competence: Order-Internal Representation vs. Hierarchical Inheritance

8.1 Knowledge as Order-Internal Representation

In a Representation Model, *knowledge* is defined relative to an Order. At a given Order, knowledge corresponds to the representational space that has reached closure under that Order's constraints. It is validated, stationary with respect to that Order, and available for reasoning.

Importantly, knowledge is not static in the absolute sense. Within an Order, the system may still explore, recombine, and reason over knowledge dynamically, provided that all operations remain admissible under the Order's constraints. Knowledge is therefore both stable and operational: it defines what is known *within* an Order and what can be legitimately computed there.

8.2 Competence as Hierarchically Inherited Constraint

Competence is a distinct concept. Competence is not produced within an Order; it is inherited by that Order from lower (inferior) Orders. It consists of stabilized representations, constraints, and invariants that have already achieved closure at those lower Orders.

For a higher Order, competence is not an object of exploration or revision. It functions as a set of non-negotiable constraints that delimit the admissible reasoning space. Competence defines what the system is not allowed to violate when operating at higher levels of abstraction.

8.3 Hierarchical Relation Between Knowledge and Competence

The same representational structure may play different roles depending on perspective:

- within the Order where it is validated, it constitutes *knowledge*;
- for higher Orders, it constitutes *competence*.

This hierarchical shift is fundamental. What is dynamically explored at one level becomes a fixed constraint at the next. In this way, competence is the mechanism by which learning becomes durable and cumulative.

8.4 Competence Stabilization and Hallucination Prevention

Hallucination can be reinterpreted structurally as a failure to preserve competence. In architectures where higher-level reasoning can overwrite lower-level regularities, competence collapses and incoherent outputs are produced.

RMs prevent this failure mode by design. Once a representation has stabilized as competence at a lower Order, it cannot be contradicted by higher Orders. Any apparent contradiction is detected as non-closure, triggering governed adaptation rather than silent violation.

8.5 Progressive Learning Through Competence Accumulation

Learning in an RM is progressive and hierarchical. Early learning phases establish lower-Order knowledge, which stabilizes into competence. This competence then serves as the foundation for higher-Order reasoning and learning.

This mechanism mirrors the developmental trajectory observed in human cognition: sensory and perceptual competencies acquired early remain stable throughout life and support, rather than being undermined by, later abstract reasoning. In RMs, this progression is enforced architecturally rather than emergent.

9 Formal Canonical Definition of the Representation Model (RM)

9.1 Definition

A *Representation Model* (RM) is an artificial computation system whose primary mode of computation operates over explicit, structured, and hierarchical representational spaces.

Formally, an RM is defined as a tuple:

$$\mathcal{L} = (\mathcal{O}, \{\mathcal{R}_N\}, \{\mathcal{C}_N\}, \{\mathcal{I}_N\}, \mathcal{D})$$

where:

- $\mathcal{O}$ is a partially ordered set of *Orders*, indexed by N, defining a hierarchy of representational and computational regimes;
- $\mathcal{R}_N$ is the representational space associated with Order N;
- $\mathcal{C}_N$ is the set of validity constraints, invariants, and admissible operations governing $\mathcal{R}_N$;
- $\mathcal{I}_N$ is the set of interfaces linking $\mathcal{R}_N$ to adjacent Orders and external modalities;
- $\mathcal{D}$ is the set of dynamical operators governing representational transformation and adaptation within and across Orders.

10 Representation Models and Coherent Reasoning

A Representation Model (RM) is not defined by a specific algorithm, software architecture, or computational substrate. Rather, it denotes a *class of reasoning systems* characterized by a common set of structural properties governing how representations are constructed, refined, stabilized, and closed.

The central objective of RM is to enable *coherent reasoning* in domains where implicit, approximate, or purely statistical inference is insufficient. In this context, coherence refers not to correctness of conclusions, but to the structural traceability and controllability of the reasoning process itself.

[Coherent Reasoning] A reasoning process is said to be *coherent* if, at every stage of inference, the system maintains explicit representations of: (i) the objects under consideration, (ii) the invariants governing their evolution, and (iii) the conditions under which reasoning is allowed to conclude (closure), refine (non-closure), or transition across representational Orders.

10.1 Canonical RM Properties (Architectural, Not Algorithmic)

We characterize Representation Models by a set of architectural properties that distinguish them from language-centric and purely statistical systems.

(R1) Explicit Representational Commitments. Representational objects used for reasoning are explicit and inspectable.

(R2) Explicit Non-Closure. Non-closure is represented as a first-class computational condition; forced completion is disallowed.

(R3) Hierarchical Organization. Representations are organized into Orders that constitute coherent computational spaces.

(R4) Constraint-Governed Validation. Representational commitments are accepted only through explicit validation against declared constraints and inherited competence.

(R5) Traceability. The system exposes the lineage of representational commitments, rejections, and unresolved conditions.

These properties do not prescribe an operational mechanism. They define what an RM must guarantee structurally, independent of programming language, execution substrate, or optimization choices.

[RM-Equivalent System] A system is said to be *RM-equivalent* if it satisfies Properties R1–R5, independently of its internal implementation, programming language, or computational substrate.

11 Positioning of Representation Models

This section positions Representation Models (RMs) relative to existing AI paradigms. The objective is not to rank systems by performance, but to clarify architectural distinctions, representational assumptions, and modes of computation.

11.1 RM and Large Language Models

Large Language Models (LLMs) implement reasoning within an implicit representational space learned through large-scale statistical optimization. This space encodes rich regularities and supports non-trivial inference, but it remains opaque, untyped, and architecturally flat. Learning, reasoning, and competence are collapsed into a single undifferentiated geometry.

Language Representation Models (LRMs) denote RMs specialized for linguistic structures. In their explicit and hierarchical form, LRMs organize representations into Orders governed by declared constraints, enabling competence stabilization and controlled reasoning beyond probabilistic continuation.

As a consequence, LRMs do not eliminate reasoning present in LLMs; they reorganize it. Reasoning becomes constrained, traceable, and governable. Non-closure is not resolved by probabilistic continuation but by explicit representational handling under declared constraints.

11.2 RM and World Models

World Models aim to construct internal representations of environmental dynamics, often focusing on physical or spatiotemporal prediction. Their primary objective is to minimize prediction error over latent state transitions, enabling planning and control in embodied or simulated environments.

RMs share with World Models the emphasis on internal representation, but diverge in optimization objective and scope. World Models remain primarily predictive: they seek to approximate the evolution of the world. RMs are constructive: they seek to resolve representational non-closure and establish stable invariants.

Moreover, RMs are not restricted to physical domains. Their representational hierarchy accommodates symbolic, normative, logical, and social structures, with explicit mechanisms for handling vagueness, paradox, and threshold sensitivity. Where World Models refine predictions, RMs refine representations.

11.3 RM and Neuro-Symbolic Approaches

Neuro-symbolic systems combine neural components with symbolic reasoning engines, often by embedding symbolic constraints into learning processes or by post-processing neural outputs with rule-based systems.

RMs differ in two key respects. First, symbolic structure is not an external add-on but an intrinsic component of the representational hierarchy. Second, constraints are not injected heuristically; they are enforced by the representational architecture itself.

Rather than training neural components to behave logically, RMs separate generation, representation, and validation. Neural models may be used within specific Orders or as interfaces, but coherence is enforced by explicit representational constraints.

11.4 RM and Logic Programming

Logic programming systems such as Prolog or Answer Set Programming solvers operate over explicit symbolic rules and provide strong guarantees within closed, well-specified domains. However, they treat vagueness, self-reference, and inconsistency as modeling errors to be avoided.

RMs are not logic programming systems. They are representational architectures in which paradox, vagueness, and non-closure are first-class computational phenomena. Rather than failing or requiring manual reformulation, RMs represent non-closure explicitly and treat it as a governed condition.

Logic programming systems can be integrated within RMs as specialized computational spaces at appropriate Orders. They are complementary tools, not competing paradigms.

11.5 Summary

Representation Models constitute a distinct architectural class. They differ from existing paradigms not by replacing learning with rules or prediction with planning, but by introducing explicit, hierarchical representation as the primary object of computation.

12 Architectural Instantiation of Representation Models

The architecture of a Representation Model is not defined by a single dual structure, but by the interaction between two foundational principles:

1. a hierarchy of *Orders*, each constituting a coherent computational space;
2. a recursive deployment mechanism that enables controlled expansion across Orders through representational adaptation.

Within this framework, each Order instantiates an internal representational organization that supports reasoning and validation. This section presents these elements in a conceptual order for exposition; it is not a normative implementation sequence.

12.1 Orders as Computational Spaces

An *Order* defines a coherent computational space characterized by:

- a class of representational objects admissible at that level;
- a geometry governing how these objects can be combined or transformed;
- validity constraints and closure conditions;
- bounded computational resources appropriate to the representational regime.

Reasoning is always local to an Order. There is no universal computational space shared across the system. Instead, different Orders correspond to different regimes of computation, each adapted to the nature and scale of the representations it manipulates.

12.2 Recursive Multi-Order Deployment: Representational Adaptation

The RM architecture is inherently recursive: computation is organized across Orders that define coherent representational and computational regimes. Orders are not merely fixed layers; rather, the active computational regime may adapt when closure cannot be achieved under the constraints of the current Order.

When reasoning fails to reach closure, the architecture supports *representational adaptation*: the system may shift to a representational regime in which the configuration becomes expressible, or apply further refinement within the current regime until closure becomes attainable.

This document states the architectural requirement for such adaptive multi-Order deployment but does not prescribe a specific operational mechanism. Concrete triggering policies, decomposition strategies, and execution procedures are implementation-dependent and intentionally outside the scope of this foundational paper.

12.3 Order-Internal Separation: Stabilization, Exploration, and Validation

Within each Order, the computational space is organized to preserve the separation between (i) *stabilized* representational content that is treated as valid at that Order, and (ii) *exploratory* computation that generates provisional candidate structures. A first-class *validation boundary* evaluates provisional structures against the stabilized content and the Order's declared constraints before any representational commitment occurs.

This separation is an architectural requirement rather than a prescription of a specific implementation.

12.4 Architectural Consequences

By separating:

- Orders as computational spaces,
- recursive deployment across Orders,
- and Order-internal separation between stabilization, exploration, and validation,

the RM architecture ensures:

- preservation of lower-Order competence;
- prevention of competence collapse during higher-level reasoning;
- explicit traceability of representational change;
- governed expansion of computational capacity.

This architecture establishes Representation Models as Glass Box systems: their reasoning, validation, and evolution are structurally explicit rather than emergent.

13 Orchestration and Multi-Clock Computation

This section describes a conceptual interpretation of multi-Order computation. It is not a normative specification of orchestration mechanisms.

Representation Models do not rely on a single global execution timeline. Instead, computation is distributed across multiple Orders, each operating within its own computational space and temporal regime.

13.1 Local Computation and Order-Relative Time

Each Order in an RM operates under its own notion of progression, corresponding to the resolution, stability, and scope of the representations it manipulates. This progression can be understood as an Order-relative clock: a local measure of when computation advances, stabilizes, or requires adaptation.

These clocks are not synchronized a priori. Lower Orders may evolve rapidly and locally, while higher Orders operate over longer horizons and broader contexts. The architecture does not impose a universal temporal frame across Orders.

13.2 A Coordination Boundary (Non-Normative)

Practical implementations may introduce a coordination boundary to manage interactions among Orders operating under distinct computational and temporal regimes. Such a boundary must not replace Order-local validation, nor bypass competence constraints.

This paper does not prescribe the existence, structure, or responsibilities of any orchestrator component. Coordination mechanisms are implementation-dependent and may be realized through multiple designs, including fully emergent coordination.

13.3 Synchronization Without Global Time

Synchronization in an RM is not achieved by aligning clocks to a common reference. Instead, it is achieved by enforcing consistency conditions between local computational states.

When interactions across Orders are required, coordination is achieved by enforcing consistency constraints between local representational states under scoped conditions. Once coherence is restored or a structural decision is taken, Orders resume independent progression.

13.4 Local Resolution and Global Coherence

A central design principle of RMs is that resolution remains local whenever possible. Most reasoning, refinement, and validation occur entirely within an Order. Cross-Order interaction is invoked only when non-closure persists beyond local resolution capacity or when inherited competence constrains higher-level reasoning.

13.5 Implications for Scalability and Multi-Interface Interaction

Because orchestration is based on synchronization rather than centralized control, RMs naturally support scalable reasoning across many Orders and simultaneous interaction with heterogeneous interfaces (physical, perceptual, cognitive, symbolic).

14 Orchestration as an Emergent Property of the Architecture

It is important to clarify that orchestration in a Representation Model is not necessarily implemented as a distinct architectural component. In a mature RM, orchestration may emerge from structural properties of the system itself. Early implementations may introduce explicit coordination mechanisms, which should be understood as provisional realizations of a deeper architectural principle.

15 Representation Models and the Foundations of Safe AGI

Representation Models are not limited to reasoning over established knowledge. By design, they constitute a recursive and self-adaptive architecture capable of identifying the limits of its own computational regime and extending it in a controlled manner.

16 Architectural Validation of Representation Models

A foundational architectural framework requires more than formal definition: it must be supported by evidence that its core principles can be instantiated and exercised in practice. This section reports

on a set of *architectural validation programs* conducted within the RCUBEAI research program, designed to test key structural properties predicted by the Representation Model (RM) framework.

The objective of these programs is not performance benchmarking or competitive comparison with existing systems. It is the validation of structural and behavioral invariants that distinguish RMs from language-centric or purely statistical architectures. Each program targets a distinct aspect of the RM definition. Together, they provide convergent evidence for the feasibility and internal coherence of the RM paradigm.

16.1 Validated Architectural Properties

Across the validation programs, the following RM properties were explicitly tested as *architectural consequences* of the framework, rather than as task-specific optimizations or heuristics:

- **Explicit representational stabilization:** the ability to distinguish exploratory computation from validated representational commitments, and to prevent premature or ungoverned commitment.
- **Non-closure detection and handling:** the explicit identification of unresolved configurations as first-class computational conditions, without resorting to forced probabilistic completion.
- **Hierarchical competence preservation:** the inheritance of stabilized constraints across computation Orders, ensuring that higher-level reasoning cannot silently violate lower-level competence.
- **Cross-space representational consistency:** the coordination of multiple computational space instantiations (e.g. linguistic, logical, normative) under a unified closure framework, with explicit handling of inter-space non-closure.
- **Auditability and traceability:** the capacity to expose the conditions under which representational commitments are accepted, rejected, or deferred, enabling post-hoc inspection without reliance on opaque probabilistic explanations.

These properties are direct consequences of the RM definition (Properties R1–R5) and do not depend on specific learning mechanisms, model sizes, optimization strategies, or execution infrastructures.

16.2 Validation Programs

Multiple independent validation programs were conducted, each focusing on a different RM property in a distinct operational context. Each program was designed to test a specific architectural characteristic of the RM framework rather than to serve as a component within a single monolithic system.

The programs are summarized below solely in terms of the architectural properties they validate. *Implementation details, operational policies, internal system names, and program-specific engineering choices are intentionally excluded from this foundational paper.*

Program I — Multi-Order Organization and Competence Preservation. This program validated the feasibility of organizing computation across multiple representational Orders with hierarchical competence inheritance. It demonstrated that stabilized representations at lower computation Orders can be structurally protected from invalidation by higher-Order exploratory

reasoning, and that persistent non-closure propagates as a governed architectural signal rather than producing incoherent output.

Program II — Auditability and Constraint Enforcement. This program focused on the enforcement of explicit validation boundaries. It validated that representational commitments can be inspected post-hoc, that violated constraints and unresolved conditions can be identified without recourse to probabilistic explanation, and that the system's reasoning lineage remains traceable across validation steps.

Program III — Cross-Space Representational Closure. This program validated that outputs produced within one computational space instantiation can be treated as provisional configurations and evaluated against constraints defined in other representational spaces. It demonstrated the decoupling of content generation from representational commitment, and confirmed that cross-space non-closure is represented explicitly and handled without forced completion.

Program IV — Structured Exploration and Stabilization Dynamics. This program validated the separation between exploratory dynamics and representational stabilization in a controlled computational environment. It demonstrated that structured exploration can proceed without collapsing into premature commitment, that stabilization occurs only when explicit closure conditions are satisfied, and that the transition between exploration and commitment is governed and traceable.

16.3 Scope and Falsifiability

The validation programs reported here are not presented as exhaustive implementations of the RM framework, nor as optimal realizations of its principles. They serve as *existence and property proofs*: evidence that systems constructed under the RM paradigm can operationalize its defining architectural properties across heterogeneous contexts.

Detailed implementation strategies, empirical performance characterization, and engineering optimization choices are intentionally excluded from this document. They will be addressed in subsequent, scope-specific publications following the phased disclosure strategy of the RCUBEAI research program.

Taken together, these validation efforts support the claim that Representation Models constitute a viable and coherent architectural class, capable of supporting controlled reasoning, competence preservation, and representational governance beyond the limits of language-centric AI systems.

Failure to exhibit these properties under instantiation would constitute a falsification of the RM architectural claims, independent of empirical performance metrics. This falsifiability criterion is stated deliberately: it establishes that the RM framework makes testable structural predictions, not merely aspirational design goals.

17 Scope, Intent, and Theoretical Grounding

This document is intended as a foundational architectural paper. Its purpose is to formally define the class of Representation Models (RMs), clarify their core principles, and position them within the broader landscape of artificial intelligence research.

It is not intended to disclose internal algorithmic implementations, optimization procedures, or engineering details of specific RM instantiations.

18 Conclusion

This paper introduced the Representation Model (RM) as a new class of computational system enforcing a foundational architectural paradigm for artificial intelligence. Rather than extending language-centric or purely statistical approaches, RMs reorganize artificial computation around explicit, hierarchical representation, controlled reasoning, and governed evolution.

This document deliberately restricts itself to architectural definition and conceptual grounding. It does not disclose algorithmic implementations or engineering details. Future work will report on validation phases, empirical behavior, and selected implementation aspects under appropriate disclosure conditions.

Metadata

- **Author:** Simone Mazzoni
- **Affiliation:** RCUBEAI Research Lab, Paris, France
- **ORCID:** 0009-0004-0400-4823
- **Title:** Representation Models (RM): Explicit Representation, Controlled Reasoning, and the Limits of Language-Centric AI
- **Version:** v1.2
- **Date:** January 2026
- **Type of Paper:** Foundational Theoretical Paper
- **Validation Status:** Issued (Author-Issued)
- **DOI:** 10.5281/zenodo.18288540
- **License:** CC BY 4.0 (text)
- **Keywords:** Representation Models, explicit representation, controlled reasoning, non-closure, hierarchical computation, AI governance

How to Cite

Mazzoni, S. (2026). *Representation Models (RM): Explicit Representation, Controlled Reasoning, and beyond Language-Centric AI.* Foundational architectural paper. Version v1.2. RCUBEAI Research Lab. License: CC BY 4.0.