Writing Down Becoming

Functional Systems Modelling and the Missing Notation Between Music and Machine Reasoning

David Slater August 2026

Robert Hercock argues that new notations change the forms of thought available to the minds that use them. This essay extends his sequence by placing Functional Systems Modelling between musical notation and a future notation for machine reasoning. FSM offers a disciplined way to represent how interacting functions produce changing system states and how the consequences of earlier occurrences remain active in the present.

Robert Hercock's essay Music, Zero, Algebra and AI begins with a deceptively simple proposition: the marks invented to represent thought eventually alter the forms of thought available to us. A notation begins as a means of recording something already understood. Through use, it becomes a cognitive instrument. It exposes relationships, transfers part of the intellectual workload into an external representation, and permits reasoning that would otherwise be too cumbersome to sustain.

Marks that change the mind

Hercock develops his argument through three transformations. Zero and positional notation made large-scale calculation practicable because position itself carried meaning. Algebra allowed relationships to be represented and transformed independently of particular quantities. Musical notation placed several coordinated activities on the page and allowed performers to realise them through time. He then asks whether artificial intelligence will require a further notation suited to structured, parallel and temporal reasoning (Hercock, 2026).

The sequence can be extended, (Figure 1). Between musical notation and Hercock's proposed notation for machine reasoning lies the emergence of executable functional representations of systems. Functional Systems Modelling, or FSM, is one such development. It combines a visual grammar of interacting functions with state-bearing metadata, transition rules, repeatable execution and preserved instantiation records. It offers a way of writing down how a system changes and how each change alters the conditions from which later activity emerges.

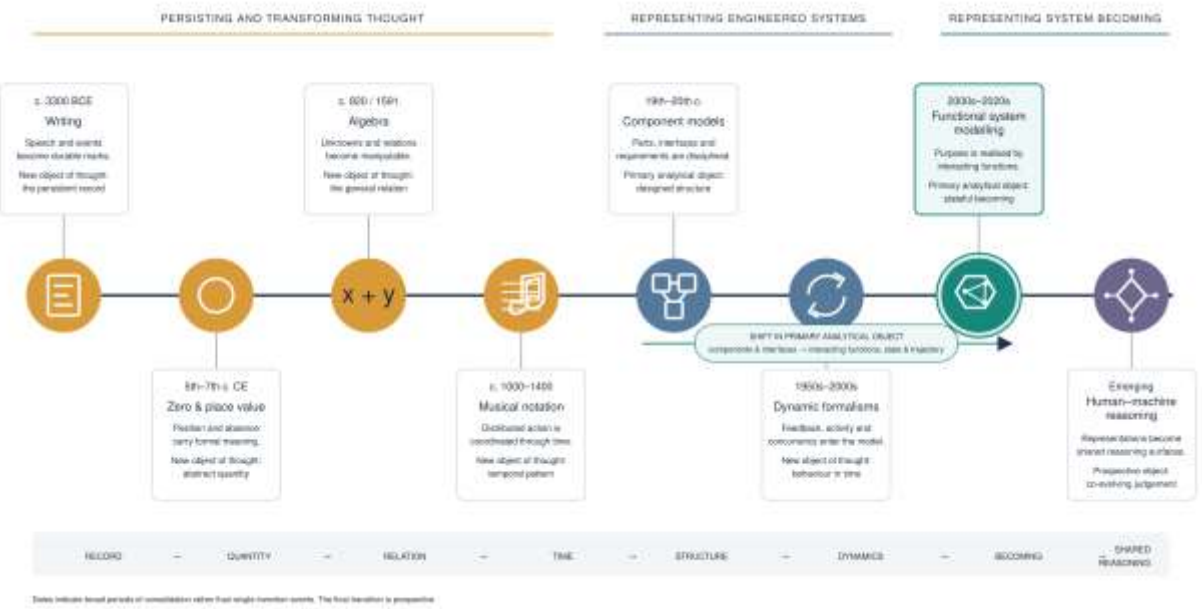

Figure 1. A proposed extension of Hercock's sequence of representational transitions. Each notation externalises a new analytical object. FSM is positioned as a contemporary bridge from component-centred representations of complicated systems to stateful representations of complex sociotechnical systems.

Functional thinking has a long lineage. Functional flow models, IDEF0 and its predecessors, state machines, Petri nets, system dynamics and discrete-event simulation already represent activity, state or change in different ways (Forrester, 1961; Harel, 1987; Zeigler, Praehofer and Kim, 2000). FSM belongs within this development. Its specific contribution lies in bringing purpose-centred functional description, context-sensitive interaction, variability, state, execution and retained evidence into a shared modelling environment.

Table 1. An extended sequence of notations and their cognitive effect

Notation	What the marks make visible	Cognitive consequence
Writing	Persistent statements	Thought can be inspected, criticised and accumulated
Positional number	Magnitude through position and explicit absence	Calculation becomes practicable at scale
Algebra	Variables, relations and lawful transformation	Families of possible cases become manipulable
Musical score	Coordinated activity arranged through time	Several performers can realise a common temporal structure
FSM	Functions, conditions, state and realised interaction	Path-dependent system behaviour becomes inspectable and executable
Machine reasoning	Structured external state and transformation	A proposed medium for reasoning about dynamic systems

Source: Extended from Hercock (2026).

From objects to functional transformations

The movement from arithmetic to algebra changed the primary object of mathematical thought. A number denotes a particular quantity. Algebra represents the relationships through which quantities may vary while remaining subject to a common structure. Once a problem can be expressed symbolically, the relationships can be manipulated directly and their consequences can be explored without reconstructing the entire argument in prose.

A comparable shift occurs when systems are represented through functions. Components denote entities within a system architecture. Functions denote activities or transformations through which a declared purpose is realised. Couplings express the relationships through which the output of one function may become an enabling or constraining condition for another. The transition concerns the centre of analysis. It moves from identifying what the system contains towards examining what must happen, what each activity requires, and how the resulting conditions affect subsequent activity.

An aircraft contains sensors, flight computers, displays, pilots, engines and control surfaces. A functional model asks how the combined system establishes the flight condition, validates air data, manages energy, configures the aircraft, maintains situational awareness and controls the trajectory. A single component may participate in several functions. A function may also be realised through several human and technical components. The observed outcome belongs to the interaction pattern and cannot be assigned automatically to one component.

Function identification remains dependent on viewpoint. The modeller declares a purpose, selects a boundary and decides which distinctions matter for the inquiry. Other stakeholders may define the purpose differently or identify competing purposes within the same setting. FSM makes these choices explicit and open to review. It does not remove the interpretive work required to construct a useful model.

What makes FSM a notation

The Functional Resonance Analysis Method, or FRAM, provides the functional grammar from which this form of FSM develops (Hollnagel, 2012). A function is described through six aspects: Input, Output, Preconditions, Resources, Control and Time. Their positions carry meaning, much as the position of a digit carries meaning in positional arithmetic. A visual coupling shows that an output from one function could satisfy a downstream aspect of another. The arrangement distributes system meaning across the page and allows functional dependence to be inspected directly.

Standard FRAM supplies a method for describing functions, couplings and performance variability. The executable FSM extension adds an operational layer. Metadata carries values and state. Guards and equations express activation and transformation rules. The runtime evaluates the current conditions, propagates outputs and updates the represented system state. A completed instantiation records which possibilities were realised under a particular set of starting conditions.

In this account, FSM is a notational system rather than a diagram alone. Its visual syntax identifies functions and typed aspects. Its semantics define what the marks and metadata mean. Its transition rules specify how the representation may change. Its execution policy determines the order in which conditions are evaluated and updates are made. Its records preserve the resulting state evidence. These elements allow the notation to support analysis, execution and comparison within the same conceptual frame.

The distinction between potential and realised coupling is central. The canonical model tables define functions, aspects, identifiers, metadata and rules. The FRAM Model Visualiser, FMV, derives and displays potential couplings from compatible outputs and downstream aspects. A potential coupling becomes a realised runtime binding only when the required output is available and the downstream activation conditions are satisfied. The realised functional organisation may therefore differ from the full field of relationships shown in the static model.

A stateful executable function within this extension has been termed a Turon. A Turon receives typed, metadata-bearing signals, evaluates its current conditions, updates its internal state and emits outputs. Its runtime bindings are late-bound because they depend on the outputs and conditions available during the instantiation. A realised system is therefore an achieved pattern of interacting functions rather than the automatic activation of every edge visible in the model.

The FRAM Time aspect represents a temporal condition affecting a function. Executable model time requires further semantics. A cycle index records the order of model evaluation, while elapsed time represents duration in the world being modelled. Delays, concurrent activation, competing updates and termination conditions require explicit rules whenever they affect the result. This separation prevents the visual presence of a Time aspect from being mistaken for a complete temporal model.

Writing down consequential time

Hercock describes musical notation as a way of writing down time. A score places several voices on the page and specifies when each should enter, how long it should continue and how it should relate to the others. FSM addresses a further property of time. An occurrence can alter the conditions under which later occurrences become possible. Each realised interaction produces a new present from which the system continues.

Occurrences matter because they leave records. Here, a record means any retained physical, organisational, cognitive or informational change through which an earlier occurrence continues to affect the present. Fuel has been consumed. Equipment has accumulated heat or wear. A queue has grown. A team has acquired information, experienced fatigue or changed its expectations. The earlier occurrence has ended, while its consequences continue to participate in the system.

An aircraft contrail offers an immediate image of this principle. It is a visible environmental trace of a previous passage, although it contains too little information to reconstruct the operational state of the flight. Fuel remaining, brake temperature, component degradation and crew workload provide stronger system examples because they influence what can happen next. The past persists operationally through the condition it has produced in the present (Slater, 2026).

This provides a practical treatment of the time paradox. Things exist in a current configuration. Occurrences happen and change that configuration. Records preserve some consequence of what happened. Models represent the states, occurrences and records relevant to the purpose of the analysis. Time enters FSM as ordered state transition, retained consequence and recoverable evidence.

Within a defined model boundary, the update can be written in a general form:

$$X(k+1) = F[X(k), U(k); \text{parameters}]$$

X(k) denotes the represented system state at cycle k. U(k) contains current influences entering the model, while the parameters express the transition rules. Earlier history, H(k), can be omitted from the next-state calculation only when X(k) retains every historical effect relevant to the next transition:

$$P[X(k+1) \mid X(k), U(k), H(k)] = P[X(k+1) \mid X(k), U(k)]$$

The resulting Markovianity belongs to the state representation relative to the chosen boundary. Fatigue, learning, hysteresis, accumulated damage, delayed effects and previous decisions must be included whenever their consequences remain active. An omitted memory effect creates an incomplete state description. The model boundary, sometimes described as a Markov blanket in this development, is an analytical boundary around the foreground functional system rather than a necessary physical wall.

Emergent behaviour can then be described with greater precision. It is a property of the realised system trajectory that cannot be attributed adequately to one function considered in isolation. The model trace shows how represented conditions and interactions combined within the model. Establishing real-world causality still requires operational evidence and an appropriate causal analysis.

The score, the performance and the record

The comparison with musical notation helps to separate specification, execution and observation. A score represents an organised field of intended activity. A performance is one realisation. A recording preserves evidence of what was played. In functional modelling, the model defines the possible functions, couplings, conditions and transition rules. Execution produces a modelled instantiation under specified initial and boundary conditions.

This distinction matters for Work-as-Imagined and Work-as-Done. A model built from procedures, design intent or expert expectations begins as a representation of Work-as-Imagined. Running it produces modelled behaviour. Evidence of Work-as-Done comes from observation, interviews, operational records or dynamic performance data. Mapping that evidence to the same functional representation allows expected, simulated and observed states to be compared without confusing one with another.

The current FRAIXL architecture separates these responsibilities. Canonical Excel tables provide the reviewable and authoritative definitions of functions, aspects, identifiers, metadata and assumptions. FMV visualises the possible functional organisation. The FRAM Model Interpreter, FMI, evaluates conditions, cycles the model and propagates state changes. FRAIXL provides the authoring, governance, recording and analysis environment, while persistent storage can retain completed states and, where cycle-level logging is enabled, the realised sequence of states. In this usage FMI means FRAM Model Interpreter and is distinct from the Functional Mock-up Interface standard used in co-simulation.

Each run is associated with a Run_ID, model version, scenario, initial conditions and timestamp. A completed whole-system metadata record is a state snapshot at a defined point, such as activation of an exit function. A trajectory requires the ordered sequence X(0), X(1), ..., X(n). Cycle-level records must therefore preserve the function states, realised bindings, values, rule results and timing information needed to reconstruct that sequence. Completed snapshots support comparison among run outcomes; they do not by themselves reveal every intermediate transition.

A persistent set of governed traces can become an experience base. Repeated instantiations may then be compared with observed operational data to identify discrepancies, recurring configurations and candidate precursors. The trace provides a model-internal account of which represented conditions enabled a function to fire and how values were propagated. It supports explanation within the model. Empirical claims about the real system require independent validation.

Complementarity with SysML v2

The useful comparison with conventional model-based systems engineering concerns modelling emphasis. SysML v2 is a general-purpose language for specifying requirements, behaviour, structure, analysis and verification. It includes actions, states and behavioural relationships, and should not be characterised as a static component notation (Object Management Group, 2025).

FSM takes the current configuration of interacting functions as its primary analytical object. It asks which potential relationships are realised under current conditions, how the participating functions change state, and what whole-system behaviour follows. SysML v2 and KerML can provide a standards-facing architecture and semantic definition. A governed projection can then select behaviour for executable functional representation. FSM execution returns state evidence, discrepancies and tested scenarios to the wider engineering environment.

The approaches therefore occupy complementary positions. Architecture identifies the entities, formal definitions and declared behaviour of the engineered system. FSM examines the realised functional organisation under specified conditions. The connection between them also offers a route for tracing observed functional behaviour back to requirements, components and verification cases.

Execution and model credibility

Execution introduces a responsibility for model credibility. Structural validation establishes that functions, aspects, identifiers and metadata are internally consistent. Semantic review tests whether domain experts accept the declared functions, boundaries and potential couplings. Execution testing verifies that the activation and transition rules operate as specified. Empirical validation compares modelled states with operational observations. A technically valid model file may still provide an incomplete account of the system, so these forms of evidence must remain distinct.

The same discipline applies to claims of emergence and explanation. Graph reachability indicates that one represented function can influence another through a chain of possible couplings. A realised trace shows which bindings occurred in the execution. Neither alone proves causal adequacy in the operational system. Sensitivity analysis, calibration, uncertainty analysis and comparison with independent observations are needed when the model is used for prediction or decision support.

A structured medium for machine reasoning

Hercock observes that current AI systems often communicate through a narrow serial channel even when their internal representations are distributed. Complex structures are flattened into text and reconstructed during each interaction. He proposes that a richer notation could hold relationships directly in its form, support lawful transformation and represent parallel processes through time.

FSM displays many of these properties as an external representation. Spatial arrangement and typed aspects express functional roles. Couplings express potential dependency. Metadata records current state. Guards and equations constrain transformations. Execution cycles produce state change. Governed traces preserve realised history. Human analysts, execution engines and AI systems can therefore inspect the same representation, even when it is exchanged through tables, structured data or an application interface.

This positions FSM as a candidate reasoning substrate rather than an internal AI notation. Its value can be tested. An AI can be asked to explain a functional state, identify anomalies, compare modelled and observed instantiations, or explore a counterfactual intervention. Performance can then be compared when the AI receives ordinary prose and when it receives the governed model and trace.

Governance remains essential. An LLM may propose candidate functions, metadata rows or interpretations. Human-reviewed canonical tables retain authority. Deterministic validation controls the executable representation, FMV and the FRAM Model Interpreter control visualisation and execution, and the resulting evidence remains available for challenge. The AI contributes interpretation and pattern

recognition within a bounded process. It does not silently rewrite the model or substitute fluent explanation for execution evidence.

The missing piece

Hercock's thesis can therefore be extended in a technically defensible way. Musical notation writes coordinated activity through time. Functional Systems Modelling writes consequential time by representing how realised interactions alter the state from which later activity proceeds. It turns system becoming into an object that can be inspected, executed and compared.

FSM belongs to an established history of functional, behavioural and dynamic modelling. Its distinctive claim is the integration of a purpose-centred functional grammar with context-sensitive binding, state-bearing metadata, explicit transition rules, repeatable execution and retained instantiation evidence. This combination provides a shared representation through which human analysts, computational engines and AI systems can examine an evolving system.

Its wider historical importance remains to be earned through formalisation, practical use and independent validation. The immediate research question is more precise: does this representation allow people and machines to reason more effectively about path-dependent system behaviour than prose, static architecture or unexecuted diagrams alone? That question can be answered through demonstrators, operational data and comparative evaluation. FSM fills a credible space in Hercock's schema because it gives the changing functional state of a system a form that can be represented, governed and tested.

References

Forrester, J. W. (1961). Industrial Dynamics. Cambridge, MA: MIT Press.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8(3), 231-274. doi:10.1016/0167-6423(87)90035-9

Hercock, R. (2026). Music, Zero, Algebra and AI: How the marks we invent to think with have always remade the mind that uses them. Essay, 5 August 2026.

Hill, R. and Slater, D. (2024). Using a metadata approach to extend the Functional Resonance Analysis Method to model quantitatively emergent behaviours in complex systems. Systems, 12(3), 90. doi:10.3390/systems12030090

Hollnagel, E. (2012). FRAM: The Functional Resonance Analysis Method. Farnham: Ashgate.

Object Management Group. (2025). Systems Modeling Language, Version 2.0. OMG SysML v2.0 specification

Slater, D. (2026). An Operational Exploration of the Time Paradox. Unpublished technical note, 28 May 2026.

Zeigler, B. P., Praehofer, H. and Kim, T. G. (2000). Theory of Modeling and Simulation. 2nd ed. San Diego: Academic Press.