MATHEMATICS / ARTIFICIAL INTELLIGENCE

The Possibility for a Machine-Native Mathematics

Most mathematics was built for human minds. If machines become serious mathematical reasoners, they may not simply prove within our languages. They may begin to invent representations, abstractions, and mathematical languages of their own.

By Komorebi P.

September 2026

Komorebi P. is a pseudonym. The point is not mystery. The author does not want identity, credentials, biography, or institutional affiliation to become proxies for evaluating the work. The pseudonym is a temporary methodological separation between the claims and the person making them, not a declaration of permanent anonymity. The work should be judged through its definitions, proofs, formalizations, code, experiments, and reproducibility before biography is allowed to become part of its interpretation.

For almost all of mathematical history, mathematics had one intended audience: other humans. Even when the subject concerned infinities, higher-dimensional spaces, or structures no one could visualize directly, its notation and concepts were shaped by human memory, human perception, and the habits of human communities. We chose definitions we could carry in our heads, symbols we could write, proofs we could inspect, and disciplinary boundaries we could teach.

That history is easy to mistake for necessity. It is not. Mathematical truth and the representations humans developed to reach it are not the same thing.

The distinction matters now because mathematics has acquired another kind of reader. Proof assistants already check arguments no individual would want to verify line by line. Computer algebra systems manipulate objects in forms chosen for execution rather than exposition. Frontier AI systems can write code,

preserve state across long trajectories, construct working models of unfamiliar environments, and search over executable objects. None of this yet constitutes a mature machine-native mathematics. But it makes the possibility difficult to dismiss.

The Komorebi program begins from a simple question: *What happens when representation itself becomes part of the mathematics?*

It asks a second question immediately after: *Once a representation has been chosen, which distinctions actually need to survive?*

The wider technical program answers those questions through papers, Lean developments, source code, computational experiments, formal certificates, and machine-readable artifacts. This article has a narrower purpose: to make the conceptual architecture legible without pretending that prose is the authoritative form of the mathematics.

THE KOMOREBI CYCLE

represent → abstract → transport → measure residual → refine → reconstruct → extract reusable law

Mathematics was built for human minds

The present form of mathematics is not inevitable. Definitions are chosen partly because humans can remember them. Notation is chosen partly because humans can read it. Proofs are organized partly according to what humans find explanatory. Subjects are divided partly because of historical development. Many intermediate concepts became canonical because they proved useful to generations of human mathematicians.

None of this diminishes mathematics. It does, however, require a distinction between

mathematical truth and the representational systems humans developed to reach it.

A mathematical object can admit many representations,

$$R_1(X), \qquad R_2(X), \qquad R_3(X), \ldots$$

and a problem that is opaque in one representation may become elementary in another. Much of sophisticated mathematics already depends on this fact. Yet the choice of representation is often treated as insight rather than as an explicit object of mathematics.

For a machine, that choice can become part of the formal search.

From mathematics in representations to mathematics of representations

Suppose an object X admits representations $R_\alpha(X)$. One may study X inside a chosen representation, but one may also study transformations

$$T_{\alpha\beta} : R_\alpha(X) \longrightarrow R_\beta(X).$$

The transformation itself then becomes mathematical data. Which invariants survive? Which operators commute with the transport? Which decompositions become visible? What information is lost? Can the source representation be reconstructed? How do transports compose?

The first Komorebi shift is therefore

$$\boxed{\text{mathematics in representations} \longrightarrow \text{mathematics of representation change.}}$$

"The difficult part of a mathematical problem may be discovering the language in which the proof becomes short."

But representation change is only part of the problem. Sometimes the important operation is not changing how an object is expressed. It is deciding which distinctions no longer need to be retained. That leads to abstraction.

Abstraction as controlled identification

Let X be a set or state space. In the simplest setting, an abstraction is a surjection

$$\alpha : X \twoheadrightarrow A.$$

It induces an equivalence relation

$$x \sim_\alpha y \quad \Longleftrightarrow \quad \alpha(x) = \alpha(y).$$

Abstraction therefore identifies distinctions. It deliberately declares some differences between mathematical states irrelevant.

The important question is not whether information was lost. Unless α is invertible, information was lost. The important question is whether information that matters was lost.

Let

$$\mathcal{F} = \{f_i : X \to Y_i\}_{i \in I}$$

be a family of tasks, observations, invariants, or mathematical questions. The abstraction α is sufficient for $\mathcal{F}$ when every task factors through it:

$$f_i = \bar{f}_i \circ \alpha.$$

Define

$$E_{\mathcal{F}} = \bigcap_{i \in I} \ker(f_i).$$

Then the elementary sufficiency criterion is

$$\boxed{\alpha \text{ is sufficient for } \mathcal{F} \quad \Longleftrightarrow \quad E_\alpha \subseteq E_{\mathcal{F}}.}$$

Every distinction destroyed by α must already be invisible to every relevant task. The informal act of abstraction becomes an exact mathematical relation.

The canonical task abstraction

The task family determines a canonical quotient

$$Q_{\mathcal{F}} = X/E_{\mathcal{F}}$$

with quotient map

$$q_{\mathcal{F}} : X \twoheadrightarrow Q_{\mathcal{F}}.$$

Formal spine: the canonical task abstraction

The quotient $q_{\mathcal{F}}$ is sufficient for every task in $\mathcal{F}$. If $\alpha : X \twoheadrightarrow A$ is any other sufficient abstraction, then there exists a unique map $h : A \to Q_{\mathcal{F}}$ such that

$$q_{\mathcal{F}} = h \circ \alpha.$$

Thus $Q_{\mathcal{F}}$ is the coarsest exact abstraction sufficient for the specified questions. The instruction “retain only what matters” has a universal mathematical form.

Abstraction is relative to possible operations

Task sufficiency alone is not enough. Two states can be indistinguishable under a current observation and become distinguishable after mathematical transformations are applied. A distinction may be latent.

Let a monoid M act on X through

$$\rho : M \to \mathrm{End}(X),$$

and let $F : X \to Y$ be an observation. Define the behavior map

$$\mathcal{B}_{\rho,F} : X \to Y^M, \qquad \mathcal{B}_{\rho,F}(x)(m) = F(\rho_m x).$$

Then

$$x \sim_{\rho,F} y \quad \Longleftrightarrow \quad F(\rho_m x) = F(\rho_m y) \quad \forall m \in M,$$

so

$$E_{\rho,F} = \bigcap_{m \in M} \ker(F \circ \rho_m).$$

The quotient

$$Q_{\rho,F} = X / E_{\rho,F}$$

retains exactly those distinctions that can become observable somewhere in the allowed operational future.

The resulting principle is stronger than ordinary task sufficiency:

a distinction is safely discardable only relative to both a question language and an operation language.

“A distinction is safely discardable only relative to both a question language and an operation language.”

The behavioral abstraction theorem

Call an equivalence relation R on X M-stable when

$$xRy \Longrightarrow \rho_m x \, R \, \rho_m y \qquad \forall m \in M.$$

Formal spine: the behavioral abstraction theorem

The relation $E_{\rho,F}$ is the greatest M-stable equivalence relation contained in $\ker(F)$:

$$E_{\rho,F} = \max\{R : R \subseteq \ker(F),\ R \text{ is } M\text{-stable}\}.$$

The canonical behavioral abstraction therefore solves a precise problem: discard as much state as possible while preserving every distinction that can affect a future admissible observation.

This formulation is deliberately domain-neutral. Different mathematical subjects may supply different objects, actions, and observations. The abstraction problem can still be the same.

Abstraction spaces and transport

Abstractions themselves form mathematical structure. Given

$$\alpha : X \twoheadrightarrow A, \qquad \beta : X \twoheadrightarrow B,$$

define $\alpha \preceq \beta$ when $\beta = h \circ \alpha$ for some map h. Equivalently,

$$E_\alpha \subseteq E_\beta.$$

Finer abstractions preserve more distinctions. Coarser abstractions identify more.

The joint abstraction

$$(\alpha \otimes \beta)(x) = (\alpha(x), \beta(x))$$

satisfies

$$E_{\alpha\otimes\beta} = E_\alpha \cap E_\beta.$$

So abstraction is not merely something done to mathematical objects. There is an algebra of abstractions themselves.

Representation change and abstraction should also be kept separate. An exact representation change

$$r : X \xrightarrow{\sim} X'$$

changes description while preserving information. An abstraction

$$\alpha : X \twoheadrightarrow A$$

changes which distinctions remain visible.

For a map $r : X \to Y$, abstraction relations admit pullback and pushforward. They satisfy the Galois

relation

$$\boxed{r_* R \subseteq S \quad \Longleftrightarrow \quad R \subseteq r^* S.}$$

Thus

$$r_* \dashv r^*.$$

When r is an exact representation equivalence, these become inverse isomorphisms of abstraction spaces. The abstraction calculus is therefore not tied to arbitrary coordinates.

Residuals: failure as mathematical data

Exact transport is not always possible. Komorebi treats failure itself as mathematical data.

Let X and Y carry operators

$$T_X : X \to X, \qquad T_Y : Y \to Y.$$

For a map $f : X \to Y$, define the structural defect

$$\partial_T(f) = T_Y f - f T_X.$$

Exact structural transport means $\partial_T(f) = 0$.

The important point is compositional.

> **Formal spine: the residual Leibniz law**
>
> For composable maps $X \xrightarrow{f} Y \xrightarrow{g} Z$,
>
> $$\boxed{\partial_T(gf) = \partial_T(g) f + g \partial_T(f).}$$

This is a genuine Leibniz law. Structural failure is not an unstructured error attached to the end of a computation. It has an algebra under composition.

For a chain

$$X_0 \xrightarrow{f_1} X_1 \xrightarrow{f_2} \cdots \xrightarrow{f_n} X_n,$$

the total defect expands as

$$\partial_T(f_n \cdots f_1) = \sum_{k=1}^{n} f_n \cdots f_{k+1}\, \partial_T(f_k)\, f_{k-1} \cdots f_1.$$

A failed transport can therefore carry provenance: where exactness was lost, how the loss propagated, and which part of the abstraction requires repair.

Refinement and abstraction geometry

An insufficient abstraction need not be abandoned. It may be refined.

Let

$$E_0 \supseteq E_1 \supseteq E_2 \supseteq \cdots$$

be progressively finer equivalence relations. For an action-observation system, define finite-horizon relations

$$E_n = \bigcap_{\ell(m)\le n} \ker(F\rho_m).$$

Then $E_{n+1} \subseteq E_n$, and the corresponding quotients form a refinement tower

$$Q_0 \longleftarrow Q_1 \longleftarrow Q_2 \longleftarrow \cdots.$$

Deeper admissible reasoning can therefore force finer distinctions.

This tower also induces a geometry. Let $\tau(x, y)$ be the least transformation depth at which an admissible observation distinguishes x and y, with $\tau(x, y) = \infty$ if no admissible operation ever does. Define

$$d(x, y) = \begin{cases} 0, & \tau(x, y) = \infty, \\ 2^{-\tau(x,y)}, & \tau(x, y) < \infty. \end{cases}$$

Then d satisfies the pseudoultrametric inequality

$$d(x, z) \le \max\{d(x, y), d(y, z)\}.$$

After quotienting by behavioral equivalence it becomes an ultrametric.

So the calculus suggests not only an algebra of abstraction, but a geometry of abstraction depth: distance measures how deeply one must reason before a distinction becomes visible.

Higher-order abstraction

Once abstractions form a mathematical space, operations on abstractions become mathematical objects themselves. An n-ary abstraction law has type

$$\Lambda : \mathrm{Abs}(X)^n \to \mathrm{Abs}(X).$$

Examples include refinement, coarsening, joint abstraction, and dynamic closure.

The hierarchy becomes

K_0 : objects,

K_1 : representations,

K_2 : abstractions,

K_3 : laws acting on abstractions,

K_4 : algebras of those laws,

K_5 : policies selecting or generating laws.

Not every operation on abstraction space should be treated as fundamental. A candidate general law should survive exact changes of representation. If

$$r : X \xrightarrow{\sim} X',$$

a representation-natural law satisfies

$$\boxed{r_* \Lambda_X(\alpha_1, \ldots, \alpha_n) \cong \Lambda_{X'}(r_*\alpha_1, \ldots, r_*\alpha_n).}$$

That gives a formal criterion for distinguishing a coordinate-dependent trick from a law of abstraction itself.

Law transport and language growth

Suppose two mathematical domains admit a translation

$$\Phi : \mathcal{C} \to \mathcal{D}$$

that induces

$$\Phi_* : \mathrm{Abs}_{\mathcal{C}} \to \mathrm{Abs}_{\mathcal{D}} .$$

If $\Lambda_{\mathcal{C}}$ is an abstraction law, one may ask whether there exists $\Lambda_{\mathcal{D}}$ such that

$$\boxed{\Phi_* \Lambda_{\mathcal{C}} = \Lambda_{\mathcal{D}} \Phi_*^n.}$$

This transports not merely a theorem or an object, but a rule for constructing useful abstractions. A transported theorem provides one consequence. A transported abstraction law can generate a family of consequences.

Mathematical history repeatedly promotes recurring procedures to named objects. A calculation becomes an operator. A symmetry becomes a group action. A decomposition becomes a transform. A universal pattern becomes a construction. A machine reasoner can, in principle, perform the same promotion deliberately.

If a reasoning pathway repeatedly uses

$$\alpha_1, \alpha_2, \ldots, \alpha_k,$$

the composite

$$\gamma = \alpha_k \circ \cdots \circ \alpha_1$$

can be promoted to a new primitive. The important transition is

$$\boxed{\text{reasoning episode} \longrightarrow \text{structural compression} \longrightarrow \text{new mathematical primitive.}}$$

That is not theorem accumulation. It is language growth.

"The system does not merely explore a mathematical space whose coordinates were fixed in advance. It participates in constructing the coordinates."

Langlands as a prototype of machine-native mathematics

The Langlands program offers perhaps the clearest classical precedent for the kind of universality being proposed here. Its power does not come from forcing arithmetic and automorphic mathematics into one canonical representation. It comes from constructing correspondences between genuinely different mathematical worlds and then demanding that carefully chosen structure survive the passage between them.

At the broadest schematic level, one encounters a pattern of the form

$$\boxed{\begin{array}{c}\text{global arithmetic data} \longrightarrow \{\text{local Galois/Weil data}\}_v \\ \longleftrightarrow \{\text{local automorphic data}\}_v \longrightarrow \text{global automorphic data}\end{array}}$$

The double arrow is classical Langlands territory, not a theorem newly supplied by Komorebi. What matters here is the architecture around it. A correspondence is useful only because it is constrained by compatibility data: Frobenius characteristic polynomials, inertia and monodromy, conductors, local factors, determinants, epsilon data, and the way local information assembles globally.

"Langlands does not make arithmetic and automorphic mathematics the same language. Its power comes from identifying structure that survives translation between them."

This is very close to the sense in which Komorebi uses the word *universal*. Universality need not mean a universal normal form. It can mean a reusable calculus for managing correspondences, tracking invariants, localizing information, recording failures of compatibility, and reconstructing global structure.

The connection is no longer only philosophical. The most recent Komorebi Papers X–XII already contain a substantial Langlands-facing test bed. Their roles are now cleanly separated: Paper X supplies the compatible-family and finite-Frobenius recovery layer; Paper XI attaches that style of representation-theoretic data to actual local-field ramification, inertia, Frobenius and conductor structure; Paper XII supplies the completed local-condition and global patching layer. The formal boundaries between these layers remain important.

Paper X is internally complete at its recorded source scopes: all 20 of its 20 live claims are checked. Its final three closures assemble three pieces that had previously remained separate. Claim 9.1 supplies the finite arithmetic separator from general Frobenius occurrence. Claim 9.2 proves closure of a supplied strictly compatible carrier under a selected finite constituent packet, carrying the common Frobenius characteristic-polynomial product, finite-copy Weil–Deligne and smooth-inertia data, local characteristic and Euler polynomials, monodromy-kernel dimensions, fixed-space trace sums, semilinear period comparisons, Hodge–Tate weight multiplicities, and every additional local condition equipped with an explicit finite-copy closure law. Claim 11.1 then composes the transported target family, the arithmetic separator, and the compatible-carrier closure into the conditional occupied-system completion theorem.[1] This completes Paper X's internal ledger; it does not assert unrestricted companion existence, carrier existence, automorphy, or a universal p-adic Hodge theorem. Those are different existence problems and remain external to what the completed internal contract proves.

[1] Paper X, final formal status, September 25, 2026: 20/20 internal claims checked at their recorded source scopes, with principal endpoint `PaperX.ConditionalOccupiedSystemCompletion.conditional_occupied_system_completion`.

Paper XI attacks a different layer: the local arithmetic carrier itself. Its developed branches construct residue and Henselian field towers, decomposition and inertia structures, Frobenius-degree pullbacks, lower ramification groups, fixed spaces, conductor summands, Swan and Artin expressions, unramified restriction laws, local polynomials, Gauss phases, and quotient-by-inertia structure. The active ramified/unramified branch records the passage

$$I = \ker(\deg), \qquad W/I \simeq \operatorname{range}(\deg),$$

and, when a degree-one Frobenius lift is available, identifies the quotient with the unramified Weil model. The subsequent local-attachment work is designed to make the tame monomial packet, the actual local-field ramification filtration, the conductor calculation, and the local polynomial refer to the same induced parameter.[2] The branch deliberately leaves class field theory, the local Langlands correspondence, and automorphic induction as classical external inputs. Epsilon/gamma matching still requires the missing additive-character and measure normalizations, and the nonmonomial wild comparison remains open.

Paper XII moves to the global assembly layer. Its revised 49-claim formal contract is internally complete: the development includes completed tensor constructions, closed ideals imposing simultaneous local conditions, local quotient factorization, support and patching statements, compatible-system machinery, and the factorization of global points precisely when the corresponding local restrictions satisfy their quotient conditions.[3] Representability, residual automorphic occupancy, applicable patching and classicality theorems, attached Galois representations, Frobenius density, semisimplification, and general Brauer–Nesbitt remain external inputs. Universal companions and unrestricted Langlands reciprocity are not claimed.

The X–XII local–global stack

The three papers can therefore be read as a typed progression rather than as three isolated applications:

Paper X: compatible-family coherence and finite-Frobenius recovery
$\Downarrow$
Paper XI: local field, inertia quotient, Frobenius, conductor, local factor
$\Downarrow$
Paper XII: local conditions, completed global tensor, patching interface.

The vertical arrows are interfaces, not claims that the entire Langlands correspondence has been proved internally. Each stage records which objects are constructed, which invariants are transported, and which classical existence theorems remain external.

The important point for this essay is therefore not that Komorebi has proved the Langlands program. It has not. The point is that once a Langlands-facing argument is made machine-operational, it naturally decomposes into precisely the objects emphasized by the representation calculus:

$$\begin{gathered}\text{local carrier} \;\to\; \text{ramification quotient} \;\to\; \text{Frobenius/monodromy data} \;\to\; \text{local invariants},\\ \text{compatible local data} \;\to\; \text{global patching conditions} \;\to\; \text{conditional global attachment}.\end{gathered}$$

Each arrow has a domain, hypotheses, preserved invariants, external inputs, and proof obligations. The mathematics begins to look less like a sequence of prose arguments and more like a typed network of

[2] Paper XI, ramified/unramified quotient v0.44.0 and Block 45 continuation records, September 17, 2026.

[3] Paper XII v0.17.0–v0.17.1 completion record, September 2026. The revised internal contract is 49/49; the separate original-wording ledger and external arithmetic inputs remain explicit.

transports.

A machine-native intermediate record for such a local component might schematically carry

$$\mathbb{L}_v = (K_v, W_{K_v}, I_v, \rho_v, N_v, \mathrm{Frob}_v, P_v, a_v, \varepsilon_v, \Delta_v, \mathsf{Cert}_v),$$

where the tuple is not proposed as a replacement for the standard theory. It is a machine-facing record of the interfaces that must agree: local field, Weil data, inertia, representation, monodromy, Frobenius, local polynomial, conductor, epsilon information, compatibility defects, and certificates. A global object would add the compatible-family spine, patching constraints, and whatever classical or newly proved bridge produces an automorphic or Galois attachment.

Seen this way, Langlands is not merely another application area. It is a prototype of the very mathematical situation a machine-native calculus must handle: several inequivalent representations of related structure, local and global views, nontrivial transport laws, invariants that must match, and sharp boundaries between what is constructed internally and what enters through an external theorem.

That also suggests a more precise relation between the two programs. Langlands seeks exceptionally deep correspondences between particular arithmetic and automorphic worlds. Komorebi seeks a calculus for representing, composing, auditing, and eventually discovering correspondence structures of that kind. The latter would not subsume the former mathematically. It would provide a machine-operational language in which such correspondence programs can be expressed and compared.

What machine-native mathematics means

The phrase *machine-native mathematics* should not mean mathematics inaccessible to humans, nor should it mean mathematics merely generated by machines. It refers to mathematics whose representational architecture is designed with machine reasoning as a first-class constraint.

Such a system may prioritize features that conventional prose does not make explicit:

- transformation contracts,
- typed invariants,
- machine-verifiable certificates,
- explicit residuals,
- composable abstraction operators,
- formal reconstruction maps,
- representation-independent structural signatures.

A human-readable theorem may remain one surface. It need not be the only authoritative surface.

The transition may already be beginning

The possibility described here is not purely hypothetical. Recent frontier systems have begun to exhibit local behaviors that resemble precursors of machine-native representation construction, though none of these observations by itself establishes a mature machine-native mathematics.

ARC Prize reported in September 2026 that GPT-6 Astra, while solving unfamiliar ARC-AGI-3 environments, repeatedly converted interaction histories into compact symbolic world models. It developed domain-specific shorthand, tracked latent state, and, in tool-enabled settings, constructed parsers, planners, search procedures, simulators, and small game-specific software libraries.[4]

A related result appeared in OpenAI's ARC-AGI-3 experiments with GPT-5.6 Sol. Retaining reasoning across actions and replacing destructive truncation with compaction substantially improved performance while reducing output-token use. The relevant point is not the benchmark number. It is that persistent compressed state allowed the model to preserve what it had learned about an environment and reuse coherent strategies over longer trajectories.[5]

Outside interactive benchmarks, AlphaEvolve provides another suggestive example. It searches over executable algorithmic objects, evaluates them automatically, retains successful constructions, and mutates them into new candidates across mathematics and computing.[6]

These systems should not be conflated with Komorebi. The point is narrower. When difficult tasks demand it, increasingly capable systems benefit from constructing intermediate languages and operational worlds that were not explicitly supplied in the original problem statement. The emerging pattern is

$$\boxed{\text{task} \longrightarrow \text{machine-constructed representation} \longrightarrow \text{reasoning} \longrightarrow \text{reusable primitive.}}$$

Komorebi asks what happens if that tendency becomes explicit mathematics rather than remaining an incidental behavior of individual systems.

Mathematics as an executable artifact

A machine-native mathematical object need not be exhausted by a prose statement. It may exist simultaneously as a theorem, a program, a proof object, a test harness, and an interface for further transformations:

$$\text{definition} \longleftrightarrow \text{implementation} \longleftrightarrow \text{formal proof} \longleftrightarrow \text{tests} \longleftrightarrow \text{transport rules.}$$

In such a system, execution is not merely an auxiliary check. The executable artifact can be one authoritative mathematical surface. A construction can be instantiated, transformed, checked against invariants, and consumed directly by another reasoning system.

This changes the role of the paper. Prose may remain the best surface for explanation, but it need not be the sole container of the mathematical object. A document can describe mathematics. An executable artifact can participate in mathematics.

[4] ARC Prize, "OpenAI's GPT-6 Astra on ARC-AGI-3," September 3, 2026. `https://arcprize.org/blog/astra`

[5] OpenAI, "How enabling two settings tripled our scores on the ARC-AGI-3 benchmark," July 29, 2026. `https://openai.com/index/how-two-settings-tripled-our-arc-agi-3-scores/`

[6] Google DeepMind, "AlphaEvolve: How our Gemini-powered coding agent is scaling impact across fields," May 7, 2026. `https://deepmind.google/blog/alphaevolve-impact/`

A mathematical intermediate representation

A useful analogy is a compiler intermediate representation. Many source languages can be translated into a common operational language, transformed there, and reconstructed into a target form.

A mathematical intermediate representation would need to carry more than syntax. Schematically,

$$\mathbb{K} = (X, R, \mathcal{F}, \mathcal{L}, \alpha, \mathcal{I}, \Delta, \mathsf{Ref}, \mathsf{Rec}),$$

where X is the mathematical system, R its current representation, $\mathcal{F}$ the task or invariant family, $\mathcal{L}$ the admissible transformation language, α the current abstraction, $\mathcal{I}$ the preserved invariants, Δ the structural defects, Ref refinement data, and Rec reconstruction data.

A transformation between such records could carry a formal certificate stating what was preserved, what was discarded, what remains approximate, and how the result can be reconstructed.

Understanding as structural compression

A proof establishes that a proposition follows. Understanding often appears when many propositions collapse into a smaller structural mechanism. In that sense, mathematical understanding is partly compression:

$$\boxed{\text{many facts} \longrightarrow \text{one transferable structural law.}}$$

A machine-native mathematics could make that process explicit. It would search not only for shorter derivations, but for representations in which families of results become instances of smaller abstraction or transport laws.

That is a different optimization target from theorem proving alone.

From theorem search to language search

Traditional automated reasoning often begins with a fixed formal language and searches for a derivation. Komorebi suggests that this may be too narrow a model of mathematical intelligence.

The larger search problem is

$$\boxed{\begin{array}{c}\text{problem} \longrightarrow \text{representation search} \longrightarrow \text{abstraction search} \\ \longrightarrow \text{language construction} \longrightarrow \text{transport} \longrightarrow \text{proof or computation}\end{array}}$$

The difficult part of a problem may not be the proof inside the current representation. It may be discovering the mathematical language in which the decisive structure becomes elementary.

The object of search therefore changes. A machine is no longer searching only theorem space. It is also searching the space of mathematical languages in which a theorem admits a useful form.

Once a successful representation, abstraction, or local operator recurs, it can be promoted to a primitive and made available to later reasoning. Language search feeds back into language growth.

Higher-order machine reasoning

Let Π_t denote a representation and abstraction policy. A reasoning episode produces evidence D_t, and the system updates

$$\Pi_{t+1} = U(\Pi_t, D_t).$$

A newly discovered abstraction law Λ_t can enlarge the law library:

$$\mathscr{L}_{t+1} = \mathrm{Cl}(\mathscr{L}_t \cup \{\Lambda_t\}).$$

The loop becomes

$$\boxed{\text{solve} \longrightarrow \text{extract structure} \longrightarrow \text{promote structure to primitive} \longrightarrow \text{change future representation search.}}$$

This is one plausible mathematical mechanism for cumulative machine reasoning.

Recursive self-improvement as a mathematical dynamical system

The connection to recursive self-improvement can be stated without requiring a machine to rewrite its own neural weights. The relevant recursion can occur in the mathematical machinery available to the reasoner.

Let a mathematical reasoning state at time t be

$$\boxed{\mathfrak{S}_t = (L_t, \mathcal{T}_t, \mathscr{L}_t, \Pi_t, \mathcal{V}_t).}$$

Here L_t is the operative mathematical language, $\mathcal{T}_t$ the certified theorem and construction library, $\mathscr{L}_t$ the library of representation, abstraction, and transport laws, Π_t the policy that selects or constructs reasoning strategies, and $\mathcal{V}_t$ the verifier or certificate discipline under which new material may be admitted.

A reasoning episode on problem data P_t produces a trace

$$D_t = \mathrm{Solve}(\mathfrak{S}_t, P_t).$$

From that trace the system may construct a candidate modification

$$\mu_t = G(\mathfrak{S}_t, D_t),$$

where μ_t may add a theorem, introduce a new abstraction law, compile a recurring reasoning path into a primitive, change the representation-selection policy, or alter the procedure that proposes later modifications. If the candidate satisfies the current certification obligations,

$$\mathcal{V}_t(\mu_t) = 1,$$

the state updates by

$$\boxed{\mathfrak{S}_{t+1} = U(\mathfrak{S}_t, \mu_t).}$$

This separates three different phenomena.

Three levels of improvement

Knowledge accumulation changes $\mathcal{T}_t$ while leaving the operative language and improvement policy essentially fixed.

$$\mathcal{T}_t \subsetneq \mathcal{T}_{t+1}.$$

Calculus growth enlarges the reusable structural machinery.

$$L_t \subsetneq L_{t+1} \qquad \text{or} \qquad \mathscr{L}_t \subsetneq \mathscr{L}_{t+1}.$$

Recursive self-improvement occurs when the update changes machinery that participates in producing future updates. In particular, the system may modify Π_t, the law-generating part of G, or another component on which later self-improvement depends.

$$(\Pi_t, G_t) \longrightarrow (\Pi_{t+1}, G_{t+1}).$$

The word *recursive* therefore refers to closure of the improvement process over part of its own future mechanism. Merely adding another theorem is not recursive self-improvement. Promoting a discovered representation into a primitive that changes future search is closer. Improving the policy that decides which

primitives to seek is closer still.

To make this quantitative, fix a problem distribution $\mathcal{P}$, a resource budget B, and a scoring functional $s(p, a)$ for candidate answers or constructions. Let $\Sigma_B(\mathfrak{S})$ be the strategies expressible by state $\mathfrak{S}$ within budget B. Define bounded capability by

$$\boxed{C_B(\mathfrak{S}; \mathcal{P}) = \mathbb{E}_{p\sim\mathcal{P}}\left[\sup_{\sigma\in\Sigma_B(\mathfrak{S})} s\big(p, \sigma(p)\big)\right].}$$

This definition is intentionally external. It evaluates what the state can accomplish, not what the state claims about itself.

A transition

$$\mathfrak{S}_t \to \mathfrak{S}_{t+1}$$

is a *certified improvement* relative to $(\mathcal{P}, B)$ when its verification obligations are preserved and

$$C_B(\mathfrak{S}_{t+1}; \mathcal{P}) > C_B(\mathfrak{S}_t; \mathcal{P}).$$

It is a *recursive* improvement when the modification responsible for that gain was generated using $\mathfrak{S}_t$ and changes a component that influences the generation of later modifications.

Conservative-extension monotonicity

Suppose an update is a conservative language extension: every strategy previously available remains available at the same or lower cost,

$$\Sigma_B(\mathfrak{S}_t) \subseteq \Sigma_B(\mathfrak{S}_{t+1}),$$

and the external scoring rule is unchanged. Then

$$\boxed{C_B(\mathfrak{S}_{t+1}; \mathcal{P}) \geq C_B(\mathfrak{S}_t; \mathcal{P}).}$$

The proof is immediate from inclusion of the feasible strategy sets. The importance of the statement is conceptual: a new abstraction primitive can be added without sacrificing prior capability if the extension is conservative. Whether it produces a strict gain remains an empirical question.

This makes clear why language growth alone is not sufficient evidence for recursive self-improvement. A larger language can be redundant, expensive, or badly searched. What matters is whether the new language changes the attainable solutions under finite resources.

The stronger recursive quantity is not capability itself but the system's ability to produce useful improvements. Define the *improvement productivity*

$$\boxed{\Gamma_B(\mathfrak{S}) = \mathbb{E}\left[\frac{\big(C_B(U(\mathfrak{S}, \mu); \mathcal{P}) - C_B(\mathfrak{S}; \mathcal{P})\big)_+}{\mathrm{Cost}_{\mathrm{imp}}(\mu)}\right],}$$

where the expectation is over modifications μ proposed by the current improvement policy and $(x)_+ = \max(x, 0)$.

A system exhibits *recursive acceleration* on this definition when successful updates increase not only task

capability but the expected efficiency of discovering subsequent improvements:

$$\boxed{\Gamma_B(\mathfrak{S}_{t+1}) > \Gamma_B(\mathfrak{S}_t).}$$

Nothing in the abstraction calculus guarantees this inequality. The process can plateau, regress, cycle, or discover improvements that are too expensive to use. The formalism identifies the claim that would have to be demonstrated rather than assuming it.

A corresponding fixed point can also be defined. Relative to a modification generator G, verifier $\mathcal{V}$, task distribution $\mathcal{P}$, and budget B, call $\mathfrak{S}^*$ *self-improvement stable* when no certified modification currently generable by the system yields a strict capability gain:

$$\boxed{\nexists \mu \in G(\mathfrak{S}^*) \quad \text{such that} \quad \mathcal{V}(\mu) = 1 \text{ and } C_B(U(\mathfrak{S}^*, \mu); \mathcal{P}) > C_B(\mathfrak{S}^*; \mathcal{P}).}$$

This is only a relative fixed point. It says that the present improvement language cannot find a better state under the stated budget and verifier. It does not imply that no better mathematical system exists.

Komorebi supplies a concrete candidate mechanism for such recursion. A reasoning trace can expose a repeated representation change or abstraction pattern; that pattern can be compressed into a law Λ_t; the law can be certified and added to $\mathscr{L}_{t+1}$; the enlarged law library can change Π_{t+1}; and the new policy can search future problems differently:

$$\boxed{\mathfrak{S}_t \to D_t \to \Lambda_t \to \mathscr{L}_{t+1} \to \Pi_{t+1} \to \mathfrak{S}_{t+1}.}$$

The recursion is therefore representational. The system improves partly by changing the mathematical coordinates, abstractions, and primitive transformations through which later reasoning is conducted.

This yields a precise connection between machine-native mathematics and recursive self-improvement:

> *A representation calculus becomes a candidate substrate for recursive self-improvement when discoveries can be certified, compressed into reusable operators, incorporated into the language of reasoning, and used to improve the process that discovers subsequent operators.*

The crucial empirical question is not whether such a loop can be written symbolically. It is whether the loop closes productively: whether a machine can repeatedly convert solved problems into certified representational machinery that measurably increases both future problem-solving capability and, in the stronger case, the efficiency with which further improvements are found.

Mathematics for humans and machines

Machine-native should not become a synonym for machine-incomprehensible. The more interesting architecture is bidirectional:

$$\boxed{\text{human mathematical representation} \longleftrightarrow \text{machine mathematical representation.}}$$

A machine may prefer a representation too large or structurally unfamiliar for ordinary human reasoning. A human may prefer a conceptual compression that hides details required for machine execution. A representation calculus can mediate between them.

The human-facing surface can emphasize meaning, structure, and explanation. The machine-facing surface can emphasize exact types, dependencies, transformations, certificates, and residuals. Neither needs to replace the other.

A useful design principle follows:

> *Machine-native representations should enlarge the representational space of mathematics without severing certified reconstruction into human-interpretable mathematical structure.*

The reconstruction need not reproduce every internal machine operation in a notation natural to people. It should expose the mathematical objects, assumptions, invariants, and consequences required to understand what was established and how it connects to existing mathematics.

Formal certainty, intelligibility, and provenance

Machine mathematics sharpens a distinction that already exists:

$$\boxed{\text{formal certainty} \neq \text{human intelligibility.}}$$

A formally verified proof can be correct without being explanatory. A beautiful human argument can be illuminating while omitting details required for machine checking. A mature mathematical system should support both.

Lean can certify derivations. Code can test computational behavior. Machine-readable structures can expose dependencies. Human exposition can explain what those structures mean. No one surface needs to carry the entire burden.

As reasoning becomes distributed across models, proof assistants, code, and transformations, provenance also becomes more than bibliography. A useful artifact may need to retain a lineage such as

$$\begin{aligned}&\text{source representation} \to \text{abstractions} \to \text{transport maps}\\ &\to \text{assumptions and invariants} \to \text{machine transformations} \to \text{formal certificate}\end{aligned}$$

A proof certificate establishes validity relative to a formal system. A provenance record explains how the certified object arose. These are different epistemic questions.

A machine-native artifact can therefore carry not only its final theorem or program, but also a typed ledger

of preserved invariants, discarded information, unresolved defects, and source dependencies.

The corpus as a mathematical object

A conventional mathematical project is usually organized around publications. A paper states definitions, proves results, and provides the principal durable record of the work.

Machine-native mathematics may invert that relationship. The primary artifact may instead be a structured corpus containing

- formal definitions and theorem statements,
- proof objects and Lean developments,
- source implementations and computational witnesses,
- tests, counterexamples, and verification records,
- dependency graphs and representation maps,
- abstraction contracts and provenance records.

A paper then becomes one projection of that object. A human reader may request the conceptual narrative. A theorem prover may request the formal dependency graph. A computational agent may request the executable realization. Another mathematical agent may request the transport and abstraction interfaces.

These are different views of one underlying research object. The future mathematical unit may sometimes be less like a paper and more like a mathematical corpus with multiple certified interfaces.

What Komorebi does not claim

Komorebi does not require the claim that all mathematics has one privileged universal representation. Nor does it require every mathematical domain to reduce to the same objects.

The proposed universality is weaker and potentially more useful. Different mathematical worlds may remain genuinely different while sharing laws governing representation, abstraction, transport, refinement, residual structure, and reconstruction.

The relevant universality is therefore not

everything is the same.

It is

> *Different mathematical worlds may admit common laws governing how structure is preserved, discarded, transported, recovered, and reorganized between descriptions.*

The research criterion

The strongest test of the program is not whether existing mathematics can be rewritten in Komorebi notation. A sufficiently expressive formalism can redescribe almost anything.

The relevant test is generative. Suppose two domains admit Komorebi representations

$$K(A), \qquad K(B).$$

Suppose a structural law is discovered in $K(A)$. Can that law be transported to $K(B)$ and produce a theorem, algorithm, representation, or computational advantage that was not explicitly inserted into the translation?

If so, the common representation contains mathematical information rather than merely common notation. That is the standard by which a representation calculus should ultimately be judged.

What would count as machine-native mathematics?

Not every mathematical artifact processed by a machine should be called machine-native. A Lean proof is machine-checkable. A computer algebra system manipulates formal expressions. A numerical program evaluates mathematical objects. A language model can generate mathematical prose. None of those facts alone implies a new mathematical language.

A stronger claim requires the representation itself to be shaped by the requirements of machine reasoning, and for that representation to create capabilities difficult to obtain from the inherited human representation alone.

At minimum, one would expect five properties:

1. **Operational explicitness.** Operations such as decomposition, transport, abstraction, refinement, and reconstruction are typed mathematical operations rather than informal instructions.
2. **Structural certification.** A transformation states its domain, preserved invariants, discarded distinctions, residuals, and reconstruction conditions.
3. **Cross-representational productivity.** A structural law can move between representations and generate consequences not explicitly encoded in the translation.
4. **Composability.** Operations compose into larger procedures while retaining provenance and correctness information.
5. **Language growth.** Successful recurring structures can be promoted into primitives available to subsequent reasoning.

A genuinely machine-native system should not merely manipulate a fixed mathematical language. Within controlled formal boundaries, it should be able to enlarge the language in which subsequent reasoning occurs.

A stronger empirical criterion

The distinction can be made operational.

Let M be a mathematical reasoning system operating in a fixed representation language L. Let M_K be an otherwise comparable system equipped with a representation and abstraction calculus capable of constructing new intermediate structures. Give both systems previously unseen mathematical problems drawn from several domains.

The relevant comparison is not merely final accuracy. One should measure proof success, computational cost, search depth, out-of-domain transfer, reuse of discovered structure, the number of reusable abstractions produced, and improvement on later problems after earlier discoveries.

Suppose at time t the system has a law library $\mathscr{L}_t$. After solving new problems, it discovers and certifies a structural operation Λ_t. Then

$$\mathscr{L}_{t+1} = \text{Closure}(\mathscr{L}_t \cup \{\Lambda_t\}).$$

If later reasoning improves because of this enlargement, the machine has not merely learned another theorem. It has improved the mathematical language available to itself.

Thus

$$\boxed{\text{learning mathematics} \neq \text{learning how to construct better mathematics.}}$$

The latter is the stronger machine-native claim.

This also gives the program a failure condition. If a Komorebi-style system cannot accumulate reusable abstractions and thereby improve later reasoning relative to an otherwise comparable fixed-language system, the strongest machine-native interpretation has not been demonstrated.

Machine-native elegance

Human mathematics has developed notions of elegance shaped partly by human cognitive economy. A proof may be admired because it is short, visual, surprising, conceptually unified, or based on a particularly natural construction.

Machines may develop somewhat different criteria. A representation could be valuable because it minimizes search branching, certificate complexity, transport cost, description length, or the cost of future abstraction. A representation that appears cumbersome to a person might be extremely economical for a reasoning system if it exposes a transformation algebra that is otherwise hidden.

This raises an unfamiliar possibility: there may be mathematical structures that are natural to machines but unnatural to humans.

The appropriate response would not be to force such structures immediately into conventional notation. It would be to ask whether they possess stable mathematical properties: invariants, composition laws, universal properties, transport theorems. If they do, they are legitimate mathematical objects regardless of whether humans would have discovered them unaided.

The problem then becomes one of translation:

$$\boxed{\text{machine-native elegance} \longleftrightarrow \text{human mathematical intelligibility.}}$$

Abstraction must be stable, not merely small

A minimal representation is not automatically a good representation. An abstraction can be compact and still be brittle: a small change in the task, the admissible operations, or the surrounding mathematical context may force the system to reconstruct distinctions it had discarded.

This suggests that machine-native mathematics needs a notion of *abstraction stability*. Let the task family vary from $\mathcal{F}$ to $\mathcal{F}'$, with corresponding canonical abstractions

$$Q_{\mathcal{F}} \qquad \text{and} \qquad Q_{\mathcal{F}'}.$$

A useful abstraction should not merely minimize present information. It should change in a controlled way when the mathematical objective changes slightly.

One may therefore seek an abstraction condition number, schematically

$$\kappa_{\mathrm{abs}} \sim \frac{\text{change in the selected abstraction}}{\text{change in task or operator specification}}.$$

A large value indicates representational brittleness. A small value indicates that nearby mathematical questions share a stable state description.

This immediately changes the optimization problem. The best abstraction need not be the coarsest exact quotient for the current task. It may be a slightly richer representation whose retained structure is reusable across a neighborhood of future tasks.

There is a complementary problem. A system that only refines and adds primitives will eventually accumulate a language too large to search efficiently. Machine-native mathematics therefore needs an explicit

forgetting operation as well as an abstraction operation. A primitive that was once useful may later become redundant, dominated, or too expensive to keep in the active search language.

The natural loop is therefore not

$$\text{discover} \longrightarrow \text{add} \longrightarrow \text{add} \longrightarrow \text{add},$$

but

$$\boxed{\text{discover} \longrightarrow \text{promote} \longrightarrow \text{reuse} \longrightarrow \text{measure value} \longrightarrow \text{compress or forget.}}$$

Recursive improvement, on this view, is recursive *restructuring*. Growth and pruning are dual operations. A mathematical language improves when it becomes more capable per unit of representational and search complexity, not merely when it becomes larger.

Representation-relative complexity

A second consequence is that mathematical difficulty should not always be treated as representation-independent.

Let P be a problem and R a representation. Write

$$C(P \mid R)$$

for the resources required to solve P when expressed through R. A representation that exposes the right invariant may make $C(P \mid R)$ dramatically smaller than it is in another representation.

But representations are not free. They have a construction cost, a description cost, and often a search cost. A machine-native complexity measure should therefore account for both the cost of finding or maintaining the representation and the cost of reasoning inside it. One possible objective is

$$\boxed{C^*(P) = \inf_R \left[C_{\text{discover}}(R) + C(P \mid R) + \lambda C_{\text{maintain}}(R) \right].}$$

This formalizes a familiar mathematical experience. Some problems are hard primarily because they are being viewed in the wrong coordinates. A clever representation does not eliminate the cost of the problem; it moves part of that cost into representation discovery.

The distinction matters for artificial reasoners. A benchmark that measures only proof search inside a supplied formalization may miss the more important capability: discovering the formalization in which the problem becomes simple.

> *A machine-native measure of mathematical difficulty should charge for both solving a problem and discovering the representation that makes it solvable.*

This also suggests a concrete notion of machine-native elegance. A representation can be preferred because it minimizes a joint cost involving description length, search branching, certificate complexity, transport cost, and expected future reuse. Human elegance and machine elegance may overlap, but they need not coincide.

From observational abstraction to causal and decision sufficiency

The action-observation formalism also suggests a broader hierarchy of abstraction.

An abstraction may be sufficient for predicting observations while still being insufficient for intervention. Two states can look identical under passive observation and yet respond differently when acted upon. If the admissible language includes interventions $a \in \mathcal{A}$, one may define an interventional equivalence by requiring

$$F(\rho_{m,a}x) = F(\rho_{m,a}y)$$

for every permitted operation and intervention.

The abstraction must then preserve not merely what can be observed, but what can be made different.

Planning introduces another criterion. Let $\pi^* : X \to A$ denote an optimal decision rule for a specified objective. An abstraction $\alpha : X \to Q$ is decision-sufficient when there exists $\bar{\pi}$ such that

$$\pi^* = \bar{\pi} \circ \alpha.$$

A stronger requirement is value sufficiency:

$$V^*(x) = \bar{V}(\alpha(x)).$$

The hierarchy is therefore roughly

$$\boxed{\begin{array}{c}\text{observational sufficiency} \subseteq \text{predictive sufficiency} \\ \subseteq \text{interventional sufficiency} \subseteq \text{decision sufficiency}\end{array}}$$

with the precise inclusions depending on the task and action languages.

This matters because mathematical reasoning itself is not passive observation. A reasoner transforms expressions, chooses lemmas, changes representations, introduces auxiliary objects, and performs interventions on the state of a proof. The correct abstraction for an intelligent mathematical system may therefore be closer to a decision-sufficient state representation than to a static compression of symbols.

The larger possibility is that the same calculus used to describe mathematical abstraction can also describe the state abstractions required for reasoning agents. That would connect machine-native mathematics to a broader theory of intelligence without reducing either subject to the other.

Mathematics whose language can change

The strongest version of the machine-native possibility can now be stated compactly.

A conventional automated mathematical system has approximately the form

$$\boxed{\text{fixed language + changing theorem set.}}$$

A more advanced system would have

$$\boxed{\text{changing language + changing theorem set.}}$$

Let L_t denote the mathematical language available at time t, and $\mathcal{T}_t$ the mathematical knowledge represented within it. Ordinary accumulation changes

$$\mathcal{T}_t \longrightarrow \mathcal{T}_{t+1}.$$

Machine-native abstraction additionally permits

$$L_t \longrightarrow L_{t+1}.$$

The processes interact. A theorem can expose a recurring operation. That operation can become a primitive. The new primitive can simplify another family of problems. Those problems can reveal another abstraction.

Hence

$$\boxed{(L_t, \mathcal{T}_t) \longrightarrow (L_{t+1}, \mathcal{T}_{t+1}).}$$

The system does not merely explore a mathematical space whose coordinates were fixed in advance. It participates in constructing the coordinates.

When would the program be complete enough to judge?

A theory this broad needs a stopping rule. Otherwise every successful result can be answered by proposing another layer of abstraction.

A reasonable first completion criterion for the Komorebi program would require substantially more than an essay or a common notation. At minimum, the following pieces should exist together:

1. a formally verified core for task-relative abstraction, canonical quotients, refinement, and action-relative behavioral equivalence;
2. a verified bridge from quotient abstractions to the existing projector and residual machinery;
3. several mathematically independent realizations of the same abstraction theorem;
4. at least one nontrivial law-level transport between domains that produces a result not inserted into the translation by construction;
5. a machine-readable intermediate representation implementing the relevant objects, invariants, residuals, reconstruction maps, and provenance;
6. a representation-search system capable of using that intermediate representation on previously unseen

problems;

7. a controlled comparison in which adaptive language growth outperforms an otherwise comparable fixed-language system under matched resources;
8. at least one self-generated primitive that is formally certified, incorporated into the operative language, and then reused to improve later held-out reasoning.

These conditions are intentionally stronger than demonstrating that Komorebi can redescribe existing mathematics. They require the calculus to become operational, generative, and independently useful.

> *The decisive question is not whether many subjects can be written in the same notation. It is whether the common abstraction machinery can move a law, preserve its obligations, and create new mathematical leverage somewhere else.*

If those criteria are met, the phrase *machine-native mathematical calculus* would describe an implemented research object rather than a programmatic aspiration. If they are not, the strongest interpretation should remain open.

The future of Komorebi mathematics

The next stage of Komorebi is not simply broader application. Its central questions are increasingly internal to the calculus itself:

- Which abstraction laws are representation-natural?
- Which residuals admit universal composition laws?
- Which abstraction spaces possess useful geometries?
- When can abstraction laws be transported between mathematical domains?
- When can a reasoning system discover new abstraction primitives automatically?
- How should exact and approximate abstraction coexist?
- Can higher-order abstraction remain formally stratified while becoming operationally reflective?
- How stable are useful abstractions under task drift, and when should a machine forget or compress previously learned primitives?
- Can representation-relative complexity predict when a change of mathematical language will produce genuine computational leverage?

These are not merely questions about applying Komorebi to another field. They are questions about the mathematical structure of Komorebi itself.

Closing

For most of mathematical history, the principal mathematical artifact was written for another human. That assumption is beginning to change.

A proof may now be written partly for a proof assistant. A computational theorem may exist partly as executable code. A mathematical object may be explored by an artificial reasoner before any human develops an intuitive picture of it. A representation may be useful primarily because it enables another machine transformation.

This does not imply the end of human mathematics. It implies the arrival of another mathematical audience.

The possibility explored by Komorebi is that this audience may eventually require mathematical languages of its own. If so, the transition will not merely concern faster theorem proving. It may alter what counts as a useful representation, what counts as an elementary operation, how mathematical knowledge is compressed, and how discoveries are transferred across domains.

The relevant question is no longer simply

Can machines do mathematics?

but

> *What happens when machines begin contributing to the creation of the languages in which mathematics itself is thinkable?*

The possibility considered here is not merely that machines will prove mathematics humans could already formulate. It is that machines may increasingly participate in determining which distinctions, representations, and primitive operations are worth having in the first place.

Seen at its most general, the subject is therefore not only a calculus of representation or even a calculus of abstraction. It is a calculus of *distinguishability under transformation*: which differences can matter, which can be forgotten, which reappear after operations are applied, and which new distinctions a growing mathematical language makes visible.

Komorebi is one attempt to take that possibility literally.