Theory and Measure of Decomposability

When Global Structure Can Be Recovered from Local Solves and Bounded Interfaces

A general theory of local solvability, interface complexity, and stable recomposition

Tamás Nagy, Ph.D.

tnagyphd@gmail.com

Draft • March 2026

Executive Summary (Non-Technical)

Many hard problems become tractable only after the right split is found. A theorem proof becomes feasible after the right lemmas are isolated. A large optimization becomes manageable after the right blocks are separated. A scientific model becomes understandable after the right variables are grouped and the right shared boundary conditions are exposed. This paper is about that hidden structural property.

The central claim is that **decomposability is not merely a heuristic choice**. It is a real mathematical property of a system, problem, or phenomenon. A system is more decomposable when local pieces can be solved, analyzed, or verified with limited dependence on the rest, and when the information that must pass between those pieces remains small, stable, and sufficient for global reconstruction.

This paper therefore tries to do three things at once. First, it gives a common object language for talking about systems, partitions, interfaces, local solvability, and recomposition. Second, it introduces a **measure of decomposability** that is not just one number, but a profile recording local difficulty, interface complexity, coupling strength, reconstruction error, and stability of recomposition. Third, it lays out constructive criteria under which solving parts plus coordinating interfaces is genuinely justified rather than only hopeful engineering.

The paper is intentionally broader than one method family. Spectral tools may later provide strong diagnostics in some domains, but the theory is not defined to be spectral. The claim is more basic: **global structure is sometimes recoverable from local solves plus bounded interfaces**, and when that is true, the degree of that recoverability can be formalized and measured.

This matters because many high-value systems live exactly in this regime. Proof systems, planning systems, scientific simulators, distributed optimization, and research workflows all depend on whether a large global object can be handled through local closure plus controlled coupling. A theory of decomposability would turn that intuition into an explicit mathematical design principle.

Short Abstract

We propose a general mathematical theory of decomposability. The motivating question is simple: when can a global problem be solved, verified, or understood through local subproblems plus

a bounded interface? We define exact and approximate decomposability relative to a partition, introduce a decomposability profile that separates local difficulty from interface burden and recomposition stability, and outline constructive criteria under which local solves recover the monolithic global solution. The paper is not restricted to spectral methods, though spectral and interface-operator views emerge as important special lenses.

Abstract

This paper develops a general theory of decomposability for structured systems, problems, and phenomena. The guiding intuition is familiar across mathematics, computation, and science: a large global object is often manageable only when it can be divided into local parts that admit separate analysis, while the dependence between parts is carried by a comparatively small interface. What is usually missing is a common mathematical language for saying when this is truly the case, how strongly it holds, and how it should be measured.

We define decomposability relative to a partition of the primitive objects of a system. A partition is useful only if three things happen simultaneously: local solvability is preserved inside each block, the interface required for global consistency remains bounded, and recomposition of local solutions is stable. This leads naturally to a decomposability profile with five components: local difficulty, interface complexity, coupling strength, reconstruction error, and recomposition stability. Exact decomposability appears as the zero-error case; approximate decomposability allows controlled distortion.

The paper's main conceptual claim is that decomposability is not reducible to problem size reduction alone. A split is valuable only when the interface between blocks is information-economical and the recomposition map is well-conditioned. This reframes many familiar distinctions: line-count versus modularity in proof systems, variable splitting versus true separability in optimization, and descriptive partitioning versus causal modularity in scientific models.

We then outline a constructive theorem program. Under exact interface sufficiency, global solutions factor through local feasible sets and a finite interface state. Under compressed-interface conditions, a low-dimensional interface summary is enough for approximate reconstruction. Under bounded coupling, iterative recomposition converges to the monolithic solution with explicit stability control. Spectral methods, graph Laplacians, and Schur-complement operators provide one important special realization of this program, but they are treated here as consequences of the more general theory rather than as its defining language.

The result is intended as a foundational object language for decomposition-aware reasoning across theorem proving, verification, optimization, planning, and scientific modeling. The longer-term goal is to turn decomposition from an art of good guesses into a measurable and eventually optimizable property.

---

1. Introduction

1.1 The central question

Many difficult tasks become feasible only after an appropriate intermediate structure is found. In theorem proving, one rarely defeats a hard target by local search alone; the real gain often comes from the right lemma, invariant, normal form, or bridge object. In optimization, variable splitting is helpful only when the induced coupling is weak enough to coordinate cheaply. In scientific modeling, a many-body or multiscale system becomes understandable only if the local dynamics can be separated without losing the global law.

These examples all point to the same deeper question:

> when can a global system be recovered from local solves plus a bounded interface?

This paper treats that question as primary. The point is not merely to propose better heuristics for splitting work. The point is to identify **decomposability** as a structural property in its own right.

1.2 Core claim

The paper's core claim is:

> A system is decomposable relative to a partition when its global feasible or solution structure factors through locally solvable components and an interface of bounded complexity, with stable recomposition.

This statement has three parts.

First, decomposability is **relative to a partition**. It is not an absolute yes/no property of a system without reference to how the system is split.

Second, decomposability is **not the same as smaller pieces**. A partition that creates many cross-block obligations may reduce block size while making the true interface burden worse.

Third, decomposability is **graded**. Exact factorization is rare but important. Approximate factorization with controlled interface cost and controlled recomposition error is often the more realistic target.

1.3 Contributions

This draft makes six contributions.

1. It introduces a common object language for systems, partitions, local state, interfaces, and recomposition.
2. It distinguishes exact, approximate, and stable decomposability.
3. It defines a decomposability profile rather than forcing everything into one scalar.
4. It separates local difficulty reduction from interface burden and coupling burden.
5. It states a constructive theorem program for factorization, compression, and stable recomposition.
6. It positions spectral criteria as one strong special case rather than the identity of the theory.

1.4 Scope and non-claims

This paper does **not** claim that every hard problem becomes easy after decomposition, that every good partition can be found efficiently, or that interface structure is always low-dimensional. It also does not claim that decomposability is purely graph-theoretic. Semantic dependencies, conservation laws, and latent coupling may all matter even when an explicit graph looks sparse.

The claim is narrower and more useful: if a system admits a partition with locally sufficient closure and bounded interface burden, then global solution, verification, or understanding can be organized around that partition, and the strength of this possibility can be measured.

1.5 Why this matters here

This theory matters because many modern systems are already decomposition-dependent in practice, even when they lack a formal language for it. Proof assistants, distributed verification engines, modular scientific workflows, agent systems, and large optimization pipelines all rely on implicit judgments about what can be handled locally and what must be propagated globally. A theory of decomposability would let those judgments become explicit, testable, and eventually optimizable.

---

2. Canonical Formalism

2.1 Measured admissibility systems

To keep the theory general while still formal, we work with a **measured admissibility system**

$$\mathcal{S} = (X, d_X, \mathcal{A}, \mathfrak{r}),$$

where:

- X is a nonempty state space,
- d_X is a metric or pseudometric on X,
- \mathcal A \subseteq X is the admissible set,
- \mathfrak r : X \to [0,\infty) is a residual functional with

$$\mathcal{A} = \{x \in X : \mathfrak{r}(x) = 0\}.$$

This language covers several common cases.

- For constraint satisfaction, \mathfrak r(x) is a constraint residual.
- For verification, \mathfrak r(x) is a proof or consistency failure score.
- For optimization, \mathfrak r(x) can be objective gap above the optimum or a KKT residual.
- For scientific modeling, \mathfrak r(x) can be model mismatch or law violation.

The point is that the global problem is always represented as:

$$\text{find } x \in X \text{ such that } \mathfrak{r}(x) = 0,$$

or, in approximate settings, as finding x with small residual.

2.2 Primitive objects and partitions

Let V be the index set of primitive objects of the system: variables, equations, theorem obligations, modules, regions, factors, or modes. A partition

$$P = \{V_1, \dots, V_k\}$$

of V determines the intended block structure of the system.

For each block V_i, let X_i be a local state space and let

$$\pi_i : X \to X_i$$

be the projection or restriction map extracting the local state of block i.

2.3 Interface data

The key object is the **interface space** B_P, together with an interface extractor

$$\beta_P : X \to B_P.$$

The value \beta_P(x) is the boundary information needed to coordinate the blocks under partition P. It may encode:

- shared boundary values,
- coupling constraints,
- exported theorem signatures,
- consensus variables,
- conserved quantities,
- or other global compatibility data.

The theory does not assume a unique canonical choice of interface. Different decompositions of the same system may require different interface objects.

2.4 Local admissibility maps

For each block i, define a set-valued local admissibility map

$$\Phi_i : B_P \to 2^{X_i},$$

where \Phi_i(b) is the set of states of block i that are locally admissible under interface condition b.

The induced compatibility set is

$$\mathcal{K}_P = \left\{ (x_1, \dots, x_k, b) \in \left(\prod_{i=1}^{k} X_i \right) \times B_P : x_i \in \Phi_i(b) \ \forall i \right\}.$$

Thus \mathcal K_P is the set of blockwise-valid configurations under a common interface state.

2.5 Recomposition

A recomposition map for the partition P is a map

$$R_P : \mathcal{K}_P \to X.$$

Given locally admissible states and interface data, R_P attempts to reconstruct a global candidate state.

It is useful to define the trace map

$$T_P : X \to \left(\prod_{i=1}^{k} X_i \right) \times B_P, \qquad T_P(x) = (\pi_1(x), \dots, \pi_k(x), \beta_P(x)).$$

The entire decomposition problem can then be read as a question about whether admissible global states factor through T_P, \mathcal K_P, and R_P.

---

3. Exact, Approximate, and Stable Decomposability

3.1 Exact decomposability

We say that \mathcal S is **exactly decomposable relative to the datum**

$$\mathfrak{L}_P = (X_i, \pi_i, B_P, \beta_P, \Phi_i, R_P)_{i=1}^{k}$$

if the following hold.

(ED1) Sound localization. For every admissible state x \in \mathcal A,

$$T_P(x) \in \mathcal{K}_P.$$

(ED2) Complete recomposition. For every z \in \mathcal K_P,

$$R_P(z) \in \mathcal{A}.$$

(ED3) Exact recovery on admissible states. For every x \in \mathcal A,

$$R_P(T_P(x)) = x.$$

Condition (ED3) is the strongest exact form. In later variants one may weaken equality to observational equivalence, but the present paper adopts exact state recovery as the default definition.

3.2 Approximate decomposability

Let \varepsilon_{\mathrm{res}} \ge 0 and \varepsilon_{\mathrm{rec}} \ge 0. We say that \mathcal S is

$$(\varepsilon_{\mathrm{res}}, \varepsilon_{\mathrm{rec}})\text{-decomposable}$$

relative to \mathfrak L_P if (ED1) holds and the following two bounds replace exact completeness and exact recovery:

(AD1) Residual control on recomposition.

$$\sup_{z \in \mathcal{K}_P} \mathfrak{r}(R_P(z)) \le \varepsilon_{\mathrm{res}}.$$

(AD2) Recovery control on admissible states.

$$\sup_{x \in \mathcal{A}} d_X\big(R_P(T_P(x)), x\big) \le \varepsilon_{\mathrm{rec}}.$$

Exact decomposability is the special case

$$(\varepsilon_{\mathrm{res}}, \varepsilon_{\mathrm{rec}}) = (0, 0).$$

3.3 Stable decomposability

Assume that each X_i is equipped with a metric d_i, and that B_P is equipped with a metric d_B. Define the product metric on

$$\left(\prod_{i=1}^{k} X_i\right) \times B_P$$

by

$$d_P\big((x_1, \dots, x_k, b), (\tilde{x}_1, \dots, \tilde{x}_k, \tilde{b})\big) = \sum_{i=1}^{k} d_i(x_i, \tilde{x}_i) + d_B(b, \tilde{b}).$$

We say that the decomposition datum \mathfrak L_P is **kappa-stable** if

$$d_X(R_P(z), R_P(\tilde{z})) \le \kappa\, d_P(z, \tilde{z}) \qquad \forall z, \tilde{z} \in \mathcal{K}_P.$$

Equivalently, R_P is \kappa-Lipschitz on the compatibility set. Stability is therefore a property of the recomposition map, not merely of the partition size.

3.4 Local solvability

To compare monolithic and blockwise work, fix an abstract cost model \mathfrak c. The exact implementation of \mathfrak c is application-dependent: wall-clock time, proof-check cost, query complexity, communication cost, or oracle calls.

We call \mathfrak L_P **locally solvable under \mathfrak c** if every local admissibility task

$$\text{solve } x_i \in \Phi_i(b)$$

is well-defined for b \in B_P and can be evaluated at materially smaller cost than the global task \mathfrak r(x)=0.

This condition does not belong to exactness itself, but it is necessary if a decomposition is to be useful rather than merely representable.

---

4. The Decomposability Profile

4.1 Profile definition

For a fixed system \mathcal S, partition P, cost model \mathfrak c, and interface complexity functional \mathfrak K, define the decomposability profile of the localization datum \mathfrak L_P by

$$\mathbf{D}(\mathcal{S}; \mathfrak{L}_P) = (L(\mathfrak{L}_P), I(\mathfrak{L}_P), C(\mathfrak{L}_P), E(\mathfrak{L}_P), \Gamma(\mathfrak{L}_P)).$$

The five components measure local burden, interface burden, coupling sensitivity, reconstruction error, and stability.

4.2 Local difficulty

Define

$$L(\mathfrak{L}_P) = \sup_{b \in B_P} \max_{1 \le i \le k} \frac{\mathfrak{c}(\Phi_i(b))}{\mathfrak{c}(\mathcal{S})}.$$

This is the worst normalized local solve burden. A partition is useful only if L(\mathfrak L_P) is substantially below 1.

4.3 Interface complexity

Define

$$I(\mathfrak{L}_P) = \sup_{x \in \mathcal{A}} \mathfrak{K}(\beta_P(x)).$$

The functional \mathfrak K may measure bit length, description length, communication burden, boundary cardinality, or another domain-appropriate interface complexity.

4.4 Coupling sensitivity

Let d_{H,i} denote the Hausdorff metric on subsets of X_i induced by d_i. Define

$$C(\mathfrak{L}_P) = \sup_{b \neq \tilde{b}} \max_{1 \le i \le k} \frac{d_{H,i}(\Phi_i(b), \Phi_i(\tilde{b}))}{d_B(b, \tilde{b})}.$$

This is the worst sensitivity of local feasible sets to interface perturbation. It is a general, non-spectral notion of coupling strength.

4.5 Reconstruction error

Define

$$E(\mathfrak{L}_P) = \max\left\{ \sup_{z \in \mathcal{K}_P} \mathfrak{r}(R_P(z)), \sup_{x \in \mathcal{A}} d_X(R_P(T_P(x)), x) \right\}.$$

Thus E(\mathfrak L_P)=0 if and only if the datum is exact in both residual and recovery senses.

4.6 Stability

Define

$$\Gamma(\mathfrak{L}_P) = \inf\{\kappa \ge 0 : \mathfrak{L}_P \text{ is } \kappa\text{-stable}\}.$$

This is the smallest Lipschitz constant of recomposition on the compatibility set.

4.7 Best profile over a partition

For a fixed partition P, there may be many localization data \mathfrak L_P. The partition-level decomposability object is therefore the Pareto frontier

$$\mathbf{D}(\mathcal{S}; P) = \mathrm{ParetoMin}\{\mathbf{D}(\mathcal{S}; \mathfrak{L}_P) : \mathfrak{L}_P \text{ admissible for } P\}.$$

This keeps the theory honest: a partition and a localization choice are not the same object.

4.8 Verification-optimal decomposability

For theorem verification and related exact-check domains, approximate semantic error is typically not acceptable. The relevant optimization problem is therefore not generic approximate decomposability, but **exact decomposability under a workload model**.

Let \mathcal S_{\mathrm{ver}} be a verification system, let \mu be a probability distribution over edit events or change sets, and let \mathfrak L_P be an exact decomposition datum for a partition P. Write

$$T_{\mathrm{recheck}}(\mathfrak{L}_P; \mu)$$

for the expected recheck cost induced by edits sampled from \mu, and let

$$U_{\mathrm{reuse}}(\mathfrak{L}_P;\mu)$$

be the expected cache or proof-reuse gain under the same workload.

Define the verification objective

$$J_{\mathrm{ver}}(\mathfrak{L}_P;\mu) = \alpha L(\mathfrak{L}_P) + \beta I(\mathfrak{L}_P) + \gamma C(\mathfrak{L}_P) + \delta\Gamma(\mathfrak{L}_P) + \xi T_{\mathrm{recheck}}(\mathfrak{L}_P;\mu) - \eta U_{\mathrm{reuse}}(\mathfrak{L}_P;\mu),$$

with nonnegative weights

$$\alpha,\beta,\gamma,\delta,\xi,\eta \geq 0.$$

We say that \mathfrak L_P^\star is a **verification-optimal decomposition datum** for \mathcal S_{\mathrm{ver}} under workload \mu if

$$\mathfrak{L}_P^\star \in \arg\min\left\{J_{\mathrm{ver}}(\mathfrak{L}_P;\mu) : E(\mathfrak{L}_P) = 0\right\}.$$

If the minimization is taken jointly over all admissible partitions and all admissible localization data, we call any minimizer an **optimally decomposable verification structure** for \mathcal S_{\mathrm{ver}}.

This definition formalizes a practical point that matters for the present project:

- the semantic exactness constraint is hard,
- the optimization target is expected verification efficiency under realistic edits,
- and the best split is the one that minimizes future recheck burden while keeping interface and coupling cost under control.

4.9 Other optimal decomposability objectives

Verification-optimal decomposability is only one objective slice. The same profile

$$\mathbf{D}(\mathcal{S};\mathfrak{L}_P) = (L, I, C, E, \Gamma)$$

supports multiple optimality notions once task-specific utility terms are added.

Let \mu denote a workload or operating distribution and let

$$U(\mathfrak{L}_P;\mu) = (U_1, \ldots, U_m)$$

be auxiliary utilities (for example reuse gain, communication savings, control margin, robustness gain, interpretability gain). Define the feature vector

$$\Psi(\mathfrak{L}_P;\mu) = (L, I, C, E, \Gamma, U_1, \ldots, U_m).$$

For any nonnegative weight vector w, define

$$J_w(\mathfrak{L}_P; \mu) = \langle w, \Psi(\mathfrak{L}_P; \mu) \rangle$$

with sign conventions chosen so that lower is better. Then each objective family is an instance of:

$$\mathfrak{L}_P^\star(w, \mu) \in \arg\min \{ J_w(\mathfrak{L}_P; \mu) : \mathcal{C}_{\text{hard}}(\mathfrak{L}_P) \},$$

where \mathcal C_{\text{hard}} encodes domain hard constraints (for verification, E=0).

Important objective instances:

- **Compute-optimal decomposability**: minimize expected wall-clock or FLOP burden under \mu.
- **Communication-optimal decomposability**: minimize interface payload and cross-block synchronization burden.
- **Coordination-optimal decomposability**: minimize coupling and iterative coordination difficulty.
- **Robustness-optimal decomposability**: minimize failure blast radius and perturbation amplification.
- **Modularity-optimal decomposability**: maximize local autonomy, edit locality, and reusable closure.
- **Inference-optimal decomposability**: minimize predictive or decision error under approximation budget.

Thus the theory is not tied to a single objective; it is an objective-parametrized optimization framework over admissible decompositions.

4.10 Pareto-optimal decomposability

When objectives conflict, scalarization is not enough. Define the feasible set

$$\mathcal{F}_P = \{ \Psi(\mathfrak{L}_P; \mu) : \mathcal{C}_{\text{hard}}(\mathfrak{L}_P) \}.$$

A decomposition datum is **Pareto-optimal** if no other feasible datum improves one objective without worsening another.

This gives the correct "full-universe" interpretation:

- scalar objectives (J_w) choose one operating point,
- Pareto frontiers expose the full trade-space.

Verification-optimal decomposability is therefore a principled special point on a broader multi-objective frontier.

---

5. Formal Theorem Program

5.1 Theorem A: Exact factorization

The first theorem is the foundational equivalence result.

> **Theorem A (Exact factorization through the interface).** Let \mathcal S = (X,d_X,\mathcal A,\mat be exactly decomposable relative to \mathfrak L_P. Then: 1. the admissible set satisfies
>
> $$\mathcal{A} = R_P(\mathcal{K}_P),$$
>
> 2. the restriction T_P|_{\mathcal A} : \mathcal A \to \mathcal K_P is injective, 3. the restriction R_P|_{T_P(\mathcal A)} is a left inverse of T_P|_{\mathcal A}, i.e.
>
> $$R_P(T_P(x)) = x \qquad \forall x \in \mathcal{A}.$$

This theorem states that admissible global structure factors through the compatibility object \mathcal K_P and the recomposition map. It is the cleanest formal expression of exact decomposability.

Proof.

We prove the three claims in order.

For the inclusion R_P(\mathcal K_P) \subseteq \mathcal A, let z \in \mathcal K_P. Since \mathcal S is exactly decomposable, condition (ED2) gives

$$R_P(z) \in \mathcal{A}.$$

Hence every element of R_P(\mathcal K_P) is admissible, so

$$R_P(\mathcal{K}_P) \subseteq \mathcal{A}.$$

For the reverse inclusion \mathcal A \subseteq R_P(\mathcal K_P), let x \in \mathcal A. By sound localization (ED1), the trace T_P(x) lies in \mathcal K_P. Therefore R_P(T_P(x)) is well-defined. By exact recovery (ED3),

$$R_P(T_P(x)) = x.$$

Thus x \in R_P(\mathcal K_P), and therefore

$$\mathcal{A} \subseteq R_P(\mathcal{K}_P).$$

Combining the two inclusions yields

$$\mathcal{A} = R_P(\mathcal{K}_P).$$

Next we prove that R_P|_{T_P(\mathcal A)} is a left inverse of T_P|_{\mathcal A}. Let x \in \mathcal A. By (ED1), T_P(x) \in \mathcal K_P, and by (ED3) we have

$$R_P(T_P(x)) = x.$$

This is exactly the statement that

$$R_P|_{T_P(\mathcal{A})} \circ T_P|_{\mathcal{A}} = \mathrm{id}_{\mathcal{A}}.$$

Finally, injectivity of T_P|_{\mathcal A} follows from the existence of this left inverse. Indeed, let x,y \in \mathcal A and assume

$$T_P(x) = T_P(y).$$

Applying R_P to both sides is legitimate because (ED1) implies both traces lie in \mathcal K_P. Hence

$$R_P(T_P(x)) = R_P(T_P(y)).$$

Using (ED3) on both sides gives

$$x = y.$$

Therefore T_P|_{\mathcal A} is injective.

This proves all three statements. \square

5.2 Theorem B: Interface compression

To state a compression theorem, let

$$Q_r : B_P \to B_P^{(r)}, \qquad L_r : B_P^{(r)} \to B_P$$

be a rank-r or complexity-r compression-lift pair for the interface space. Define the interface distortion by

$$\delta_r = \sup_{x \in \mathcal{A}} d_B(L_r Q_r \beta_P(x), \beta_P(x)).$$

The target statement is:

> **Theorem B (Compressed interface reconstruction).** Assume: 1. the local admissibility maps \Phi_i are C(\mathfrak L_P)-Lipschitz in Hausdorff distance, 2. the recomposition map R_P is \Gamma(\mathfrak L_P)-stable, 3. a local selector exists for the compressed interface. Then the reconstruction built from the compressed interface has global error bounded by a function of C(\mathfrak L_P), \Gamma(\mathfrak L_P), and \delta_r.

The exact sharp bound depends on the selector model. The conceptual point is that interface compressibility can be converted into approximate decomposability.

5.3 Theorem C: Stable iterative recomposition

Suppose that for each block there exists a local solve map

$$s_i : B_P \to X_i, \qquad s_i(b) \in \Phi_i(b).$$

This induces an interface update map

$$G_P(b) = \beta_P\big(R_P(s_1(b), \dots, s_k(b), b)\big).$$

The intended theorem is:

> **Theorem C (Contraction-based recomposition).** If G_P is a contraction on B_P, then: 1. there exists a unique fixed point b^\star, 2. the iteration b^{(n+1)} = G_P(b^{(n)}) converges to b^\star, 3. the recomposed state
>
> $$x^\star = R_P(s_1(b^\star), \dots, s_k(b^\star), b^\star)$$
>
> is the unique globally consistent state generated by the local solve family.

This theorem is the bridge from structural decomposability to actual decomposition algorithms.

5.4 Interpretation

The three theorems play different roles.

- Theorem A says when decomposition is exact.
- Theorem B says when the interface can be compressed.
- Theorem C says when local solving plus interface coordination converges.

Together they give a formal route from ontology to approximation to computation.

---

6. Spectral and Interface-Based Lenses

The theory is not defined to be spectral, but spectral methods provide an important special lens once a dependency or influence graph has been constructed.

Let G = (V, E, W) be a weighted dependency graph induced by a partition candidate. Then:

- low-conductance cuts suggest good decomposition boundaries,
- Laplacian eigenvectors reveal coherent weakly coupled clusters,
- Schur complements compress interior structure into interface operators,
- singular-value decay of interface operators signals compressibility,
- normalized coupling spectra indicate whether iterative recomposition is stable.

In this view, spectral tools do not define decomposability. They diagnose one important family of decomposable regimes.

---

7. Canonical Examples

7.1 Theorem verification

In proof systems, a block may be a file, theorem cluster, or lemma family. The interface consists of imported statements, exported signatures, and dependency-sensitive proof obligations. A good split is not one that merely shortens files, but one that minimizes cross-cluster obligations while preserving local checkability.

7.1.1 Worked example: the `MassActivation` chain

The MassActivation family in this repository gives a concrete verification example. The relevant split is:

- MassActivationCoordinates.lean — coordinate and linearization layer,
- MassActivationRemainderBound.lean — stable-basis and remainder-bound layer,
- MassActivationSpectralCore.lean — perturbative and spectral-core assumptions,
- MassActivationMainTheorem.lean — capstone closure.

The import structure is essentially linear:

$$\texttt{Coordinates} \to \texttt{RemainderBound} \to \texttt{SpectralCore} \to \texttt{MainTheorem}.$$

This is decomposition-favorable for three reasons.

First, the direction of dependence is mostly one-way. The capstone file consumes the earlier layers rather than mutually entangling with them.

Second, the local burden drops sharply. The plan-level monolith proxy for the original source object is about 51 theorem declarations, whereas the post-split blocks are in the 9—14 theorem range.

Third, the interface appears semantic rather than accidental. The boundaries align with recognizable roles: construction, bound, core assumption, and closure.

In the language of this paper, this means:

- L(\mathfrak L_P) decreases because local verification units are smaller,
- I(\mathfrak L_P) remains moderate because the interface is carried by named assumption-level objects,
- C(\mathfrak L_P) appears limited because the dependence pattern is chain-like rather than mesh-like,
- E(\mathfrak L_P) must still be held at 0 by the semantic verification gate,
- \Gamma(\mathfrak L_P) is expected to remain acceptable because recomposition is mostly upstream accumulation rather than unstable cyclic coordination.

This example is useful because it shows that the theory is not merely classificatory. It yields an actual decision rule:

> favor splits in which a large theorem object separates into semantically distinct layers with mostly one-way dependence and a bounded assumption interface.

7.2 Optimization and control

In optimization, blocks correspond to variable groups or subprograms. The interface consists of shared multipliers, boundary conditions, consensus variables, or coupling constraints. Decomposability then measures whether local subproblems plus coordination recover the monolithic optimum with acceptable stability and cost.

7.3 Scientific phenomena

For scientific systems, blocks may be subsystems, scales, regions, or latent factors. The interface consists of fluxes, conserved quantities, matching conditions, or summary statistics. Here the theory asks whether a phenomenon has a real modular structure or only an observationally convenient partition.

---

8. Research Program

This paper should eventually deliver:

1. canonical definitions of exact, approximate, and stable decomposability,
2. a reusable decomposability profile,
3. factorization, compression, and stability theorems,
4. algorithms for estimating good partitions,
5. benchmarks on proof systems, verification pipelines, and optimization problems.

The broader ambition is to make decomposition a measurable object rather than a purely intuitive design choice.

---

9. Immediate Next Technical Steps

The next manuscript wave should sharpen four items.

1. Choose one canonical ambient formalism for the main theorem statements.
2. Decide whether interface sufficiency is defined categorically, information-theoretically, or operator-theoretically.
3. Write the first nontrivial approximation theorem, likely the compressed-interface reconstruction result.
4. Add one worked case study, likely theorem verification or modular proof checking, to anchor the abstract theory.

Appendix A. Lane Closure Summary (Session Record)

This appendix records the concrete outputs of the decomposition lane so the current state is preserved in the paper artifact itself.

A.1 Theory outputs completed

- Defined a canonical formalism for decomposition around:
  - measured admissibility systems,
  - partition-dependent localization data,
  - interface state and compatibility sets,
  - recomposition operators.
- Defined exact, approximate, and stable decomposability.
- Defined a decomposability profile:
  - L (local burden),
  - I (interface complexity),
  - C (coupling sensitivity),
  - E (reconstruction error),
  - Gamma (stability).
- Proved Theorem A (exact factorization through interface).

A.2 Objective framework outputs completed

- Defined verification-optimal decomposability as an exactness-constrained optimization problem under workload.
- Extended to objective-parametrized optimization over the same decomposition profile and utility extensions.
- Added explicit non-verification objective families:
  - compute-optimal,
  - communication-optimal,
  - coordination-optimal,
  - robustness-optimal,
  - modularity-optimal,
  - inference-optimal.
- Added Pareto-optimal decomposition framing as the multi-objective umbrella.

A.3 Practical implementation outputs completed

Implementation now exists in the repo (first-pass production surface):

- tools/nous/verify_model.py
  - split analysis objects and scoring,
  - intra-file dependency edge inference,
  - split metric computation and recommendation logic.
- tools/nous/gym_checker.py
  - split analysis helper surface,
  - split benchmark JSONL logging.
- tools/nous/cli.py
  - operational command:
    - ./nous gym verify split −score ...
  - JSON and human-readable output modes.

Operational benchmark log path:

- .cursor/state/logs/verify_split_scores.jsonl

A.4 Repository-grounded worked example completed

- Added and validated a concrete worked example around the MassActivation split chain:
  - Coordinates −> RemainderBound −> SpectralCore −> MainTheorem
- The example is used as a positive template for weakly coupled, semantically layered verification decomposition.

A.5 Validation status at lane close

- Targeted split-analysis and CLI-path tests are passing.
- The lane is no longer concept-only; it now has:
  - formal definitions,
  - a proved foundational theorem,
  - objective-family derivation,
  - and executable split scoring surfaces.