# UNNS Octad Operators
# Branching, Merging, Shadowing, Projection
# (Operators 5–8): Theory, Properties, Algorithms, and Examples

UNNS Research Notes

September 23, 2025

**Abstract**

We extend the UNNS operational grammar (the Tetrad: Inletting, Inlaying, Repair & Normalization, Trans–Sentifying) by introducing four additional, complementary operators — *Branching (Fission)*, *Merging (Fusion)*, *Shadowing*, and *Projection* — forming an *Octad* of UNNS operators. This document provides TeXPage-friendly formal definitions, mathematical properties, stability lemmas, algorithmic recipes, illustrative examples, and application notes (including links back to dark-matter / dark-energy and cosmogenesis hypotheses). The aim is to give a rigorous yet usable theoretical basis so these operators can be implemented, tested, and used in the UNNS corpus and interactive explorers.

## Contents

## 1 Introduction

The original UNNS grammar (Inletting, Inlaying, Repair & Normalization, Trans–Sentifying) provides a compact toolkit to couple external data to the substrate, embed internal motifs, stabilize recursive dynamics, and export invariants into perception. In many domains we require additional primitives that control divergence/convergence of recursive branches, manage hidden/latent structures, and map between representational layers. We propose four natural operators:

**5. Branching (Fission) 6. Merging (Fusion) 7. Shadowing 8. Projection**

These complete an *octad* that allows UNNS to express generative divergence, synchronization, latent constraints, and model-to-observable reduction. Below we formalize each operator, state basic properties and lemmas, provide algorithmic recipes and pseudocode, and show diagrams and examples.

## 2 Preliminaries and notation

We reuse basic UNNS notation:

- $\mathcal{U}$ — the UNNS substrate (a data structure holding nested recurrences, labeled meshes, motifs).

- $u_n(x)$ — value at discrete time/index $n$ and location $x$ (or entry in a motif).
- $\mathcal{M}(\cdot)$ — measurement / observable map from $\mathcal{U}$ to an observable space.
- $\mathcal{I}, \mathcal{J}, \mathcal{R}, \mathcal{T}$ — previously defined Inletting/Inlaying/Repair/Trans–Sentifying.
- $\mathcal{B}, \mathcal{M}g, \mathcal{S}, \Pi$ — Branching, Merging, Shadowing, Projection (operators introduced here).

We assume $\mathcal{U}$ admits local windows (finite neighborhoods) where linearization using companion matrices makes sense. Spectral radius of a local companion matrix is denoted $\rho(\cdot)$.

## 3 Operator 5: Branching (Fission)

### 3.1 Definition

**Definition 3.1** (Branching / Fission)**.** *Let $P \subset \mathcal{U}$ be a motif (finite subsequence, stencil, or local patch). A* branching operator *with branching targets $\{L_1, \ldots, L_k\}$ produces a set of one or more copies (possibly transformed) of $P$ and inserts them into target locations:*

$$\mathcal{B}_{P \to \{L_i\}}(\mathcal{U}) \;=\; \mathcal{U}' \quad \textit{where} \quad \mathcal{U}'|_{L_i} \leftarrow \mathcal{T}_i(P),\; i = 1, \ldots, k.$$

*Each $\mathcal{T}_i$ is an optional transformation: scaling, phase-shift, time-shift, ring-projection.*

Branching models duplication, fission, and the generation of parallel recursive streams.

### 3.2 Properties and intuitions

- **Degrees of freedom:** Branching increases the local degrees of freedom (DOF) by adding motifs. If motifs evolve independently, DOF grows additive in $k$.
- **Interaction effects:** If branches couple (via shared boundary conditions or coupling operators), emergent collective dynamics can arise (synchronization, beating).
- **Entropy/complexity:** Branching typically increases descriptive complexity and, under generic perturbations, can increase local Lyapunov exponents.

### 3.3 Local spectral lemma (sketch)

**Lemma 3.2** (Branching amplifies unstable modes (sketch))**.** *Consider a linearized local model where motif $P$ has companion matrix $C_P$ and spectral radius $\rho_P$. If branching creates $k$ decoupled copies then the composite local propagation operator has spectral radius $\rho_{composite} = \rho_P$ (multiplicity $k$). If the branches interact constructively via a rank-1 coupling of strength $\epsilon > 0$, then (to first order) the leading spectral radius increases by $O(\epsilon k)$.*

*Sketch.* Decoupled copies produce block-diagonal companion matrix with repeated spectrum. A small coupling adds low-rank perturbation; standard perturbation theory (Weyl / Bauer-Fike estimates) shows the leading eigenvalue shifts by an amount proportional to $\epsilon k$ times overlap factors. $\square$

### 3.4 Algorithmic recipe (Branching)

1. **Select motif** $P$ (stability/utility heuristics).
2. **Choose targets** $\{L_i\}$ (empty regions or regions flagged for augmentation).
3. **Decide transforms** $\mathcal{T}_i$ (scale, project, time-shift).
4. **Simulate locally** — test interactions and compute spectral diagnostics.
5. **Commit** if diagnostics within thresholds else adjust transforms or reject.

### 3.5 Examples

**Example 3.3** (Parallel echo trees). *A motif producing a self-similar echo can be branched into several trees; auditory sonification reveals interleaved rhythmic patterns.*

### 3.6 Applications

Branching is useful for synthetic generation of multi-scale structures, creating test ensembles, modelling cosmological bubble nucleation analogues, or spawning parallel computational tasks within the substrate.

## 4 Operator 6: Merging (Fusion)

### 4.1 Definition

**Definition 4.1** (Merging / Fusion). *Given two or more motifs $P_1, \dots, P_m \subset \mathcal{U}$ (possibly at distinct locations), a* merging operator

$$\mathcal{M}g_{(P_1,\dots,P_m)\to L}(\mathcal{U}) = \mathcal{U}'$$

*produces a fused motif at a target location $L$. The fusion is prescribed by a merging rule $\Phi$ that resolves conflicts (averaging, prioritized overwriting, synchronization projection).*

Merging models synchronization, consensus formation, and coherent aggregation.

### 4.2 Mathematical formalization

Suppose each motif is described by local state vectors $v_i \in \mathbb{R}^n$. A common merging rule is the convex average:

$$v_{\text{fused}} = \sum_{i=1}^{m} w_i v_i, \qquad w_i \geq 0, \ \sum_i w_i = 1.$$

More generally, merging may solve a least-squares fit to reconcile incompatible boundary conditions:

$$v_{\text{fused}} = \arg\min_v \sum_i \|A_i v - b_i\|^2 + \lambda \|v - v_0\|^2,$$

where $A_i, b_i$ encode contextual constraints and $v_0$ is a prior.

### 4.3 Stability lemma (synchronizing merge)

**Lemma 4.2** (Merging reduces DOF and can be contractive)**.** *Let the merging rule be convex averaging. Then the fused state lies in the convex hull of inputs; distances to the centroid are reduced. If inputs differ by bounded noise, merging averages out zero-mean fluctuations, acting as a contractive operator in expectation.*

*Proof.* Immediate from properties of convex averages: $\|v_{\text{fused}} - \bar{v}\| \leq \max_i \|v_i - \bar{v}\|$, where $\bar{v}$ is the centroid. □

### 4.4 Algorithmic recipe (Merging)

1. **Select inputs** $P_1, \dots, P_m$ and target $L$.
2. **Choose merging rule** $\Phi$ (average, weighted, fit).
3. **Simulate fused dynamics** locally; compute residue reduction.
4. **Accept/reject** based on stability metrics.

### 4.5 Examples

**Example 4.3** (Phase synchronization)**.** *Two oscillator motifs merged using a phase-locking rule produce a single coherent oscillator at $L$.*

### 4.6 Applications

Merging is essential for data-fusion in UNNS (combining observations), for topology engineering (gluing cells), and for modeling coalescence phenomena (galaxy mergers, synchronized neural assemblies).

## 5 Operator 7: Shadowing

### 5.1 Definition

**Definition 5.1** (Shadowing)**.** *A* shadowing operator $\mathcal{S}_{h,\chi}$ *inserts (or reveals) a latent or "shadow" field $h$ into the substrate that influences internal dynamics but is not directly observable through a given measurement map $\mathcal{M}$. Formally, with $\mathcal{M} : \mathcal{U} \to \mathcal{O}$, $h$ satisfies $\mathcal{M}(\mathcal{U} \oplus h) = \mathcal{M}(\mathcal{U})$ while dynamics $\mathcal{D}$ on $\mathcal{U}$ are altered: $\mathcal{D}(\mathcal{U} \oplus h) \neq \mathcal{D}(\mathcal{U})$.*

Shadowing formalizes hidden layers: coefficients or motifs that reside in the kernel of the observable map but affect deeper dynamics (e.g. gravitationally relevant but electromagnetically invisible fields).

### 5.2 Linear observability lemma

**Lemma 5.2** (Shadow fields lie in measurement kernel)**.** *Let $\mathcal{M}$ be linear and represented by matrix $M$ on finite-dimensional representations. A shadowing field $h$ produces no measurable change iff $Mh = 0$. Conversely, any $h$ in $\ker M$ can be used as a shadow insertion.*

*Proof.* Direct linear algebra: $\mathcal{M}(\mathcal{U} + h) = M(\mathcal{U} + h) = M\mathcal{U} + Mh$. For no measurable change we require $Mh = 0$. □

### 5.3 Physical interpretation and dark-sector link

Shadowing is the formal UNNS primitive for "hidden sectors": degrees of freedom that do not couple to certain interactions (e.g. electromagnetic) but influence others (e.g. gravitational). In the UNNS dark-matter hypothesis, hidden inlayed coefficients $h^{(\ell)}$ are a form of shadowing.

### 5.4 Algorithmic recipe (Shadowing)

1. **Specify measurement map** $\mathcal{M}$ (which observables are accessible).
2. **Construct candidate shadow fields** $h$ (e.g., motifs projected to $\ker \mathcal{M}$).
3. **Test dynamics** with and without $h$ to understand indirect effects.
4. **Calibrate** hidden fields to match indirect observables (e.g. gravitational signatures).

### 5.5 Examples

**Example 5.3** (Hidden coefficient halos)**.** *Add an inlayed coefficient field h on an outer lattice such that electromagnetic observables unchanged but effective potential modified; use to model halo effects.*

### 5.6 Caveats

Shadowing can be abused to "explain anything" unless constrained by falsifiable indirect consequences. Good priors and parsimony are required.

## 6 Operator 8: Projection

### 6.1 Definition

**Definition 6.1** (Projection)**.** *A* projection operator $\Pi : \mathcal{U} \to \mathcal{V}$ *reduces the UNNS substrate (possibly high-dimensional or richly labeled) to a lower-dimensional representational or observable space* $\mathcal{V}$*. Projections may be linear (matrix* $\Pi$*) or nonlinear (manifold maps), and can be designed to preserve chosen invariants.*

Projection covers mapping to physical observables (e.g. spatial fields), to perceptual channels (Trans–Sentifying), or to compressed representations for analysis.

### 6.2 Invariant-preserving projection lemma

**Lemma 6.2** (Commuting projection preserves recursion eigenmodes)**.** *Let* $C$ *be a linear recurrence operator on* $\mathcal{U}$*, and* $\Pi$ *a linear projection. If there exists* $C'$ *on* $\mathcal{V}$ *such that* $\Pi C = C'\Pi$ *(commutation), then eigenvectors of* $C$ *mapped by* $\Pi$ *are eigenvectors of* $C'$ *(possibly collapsing multiplicities). Thus characteristic growth rates (eigenvalues) that commute with* $\Pi$ *are preserved in projection.*

*Proof.* If $Cv = \lambda v$ then applying $\Pi$ gives $\Pi C v = \lambda \Pi v$; with $\Pi C = C'\Pi$ we have $C'(\Pi v) = \lambda(\Pi v)$, so $\Pi v$ is eigenvector of $C'$. □

### 6.3 Algorithmic recipe (Projection)

1. **Select target space** $\mathcal{V}$ and desired invariants to preserve.
2. **Design/choose** $\Pi$ (PCA, spectral projection, DEIM, coarse-graining).
3. **Analyze commutativity** with recursion operators; if not commuting, quantify distortion.
4. **Render or store** projected representation.

### 6.4 Examples

**Example 6.3** (Coarse-grain to continuum)**.** *Project fine lattice UNNS data to a smooth density field using convolution / mollifier* $\Pi$*, enabling Poisson solves and coupling to gravity.*

### 6.5 Applications

Projection is central for visualization, data-export, trans-sentifying, and coupling UNNS discrete models to continuum PDEs.

## 7 Algebra of operators and interactions

Operators in the octad may be composed. Important structural points:

- **Non-commutativity:** In general $\mathcal{B} \circ \mathcal{M}g \neq \mathcal{M}g \circ \mathcal{B}$. Example: branching then merging different copies differs from merging then branching the fused motif.
- **Associativity:** Composition is associative when operators are well-defined maps on $\mathcal{U}$ (function composition), but domain issues arise when operations alter admissible targets.
- **Dualities:** Projection is (in a loose sense) dual to inlaying: inlaying embeds into a richer space; projection reduces to a simpler one. Shadowing is dual to trans-sentifying with respect to observability.

**Proposition 7.1** (Simple non-commutativity example)**.** *Let $P$ be motif and $L_1, L_2$ two targets. Then*

$$\mathcal{M}g_{(\mathcal{B}_{P\to\{L_1,L_2\}}(P))\to L} \neq \mathcal{B}_{\mathcal{M}g_{(P,P)\to L}(P)\to\{L_1,L_2\}},$$

*except in special symmetric cases.*

*Sketch.* Left-hand side: first create two copies then merge them; right-hand side: first merge the motif with itself then branch the merged motif. Unless merging is idempotent and branching is linear with identical transforms, outcomes differ (local arrangement, phases). □

### 7.1 Operator algebra use-cases

Design rules and safety guards can be specified in terms of allowed compositions and thresholds. For instance, avoid $\mathcal{B}$ followed by $\mathcal{B}$ inside a region unless spectral radius is below limit.

## 8 Stability and complexity analysis

We provide heuristic statements and lemmas showing how new operators affect UNNS stability.

**Lemma 8.1** (Branching–Merging tradeoff)**.** *Branching increases combinatorial complexity (number of possible trajectories), while merging reduces it. A policy alternating branching and merging can maintain controlled complexity while enabling exploration.*

*Sketch.* Count DOF: branching increases DOF roughly by factor $k$; merging reduces DOF toward average. Complexity measures (Kolmogorov, Shannon) grow with branching; merging compresses. $\square$

**Proposition 8.2** (Shadowing identifiability)**.** *Shadow fields inside* $\ker \mathcal{M}$ *are not identifiable from* $\mathcal{M}$ *alone; only indirect effects (e.g. dynamical influence measurable via different observables or via responses to probes) can reveal them.*

## 9 Numerical experiments and implementation notes

We sketch practical tests and pseudocode for prototyping.

### 9.1 Prototype pseudocode: branching + merging

```
function branch_and_merge(U, motif P, targets Ls, mergespec Phi):
  U1 = B_{P -> Ls}(U)   # branch copies
  for each local region R influenced:
    compute local companion C_R
    if rho(C_R) > rho_max:
      # apply merging or repair to stabilize
      U1 = MergingRepair(U1, R, Phi)
  return U1
```

### 9.2 Shadowing calibration (inference)

1. Define measurement operator M and dataset Y (observations).
2. Propose parametric hidden field $h(\theta)$.
3. Simulate dynamics with $h(\theta)$ and compute indirect observables $\tilde{Y}(\theta)$.
4. Fit $\theta$ minimizing loss $\|Y - \tilde{Y}(\theta)\|^2$ subject to prior on $\theta$.

### 9.3 Projection selection

Common projection choices:

- PCA / SVD on local windows.
- Proper orthogonal decomposition (POD).
- Discrete Empirical Interpolation (DEIM) for nonlinear reductions.
- Mollifier convolution for coarse-graining to continuum fields.

## 10 Conclusion

Expanding UNNS from a tetrad to an octad equips the substrate with primitives to express divergence (branching), coherence (merging), hidden constraints (shadowing), and model reduction (projection). These operators are natural, interoperable, and directly relevant to the theory and applications we are building (from mesh-based FEEC experiments to speculative cosmological models). This document provides a practical, formal basis for implementation, testing, and further mathematical development.