Operator-to-UNNS Mapping: A Structural Reference

UNNS Research Notes

Abstract

This note provides a reference mapping between concrete operator implementations (as used in the *UNNS vs Classical Dual Sequence Explorer*) and the abstract grammar of the UNNS substrate. Each operator is interpreted in terms of its theoretical role in the recursive architecture: Collapse, Inlaying, Inletting, Adoption, Evaluation, and Normalization.

1 Operators as Substrate Actions

Definition 1 (Operator). *A* UNNS operator *is a functional transformation*

$$\mathcal{O} : \mathbb{C} \longrightarrow \mathbb{C}$$

applied to each step of a recursive process, modifying its trajectory while respecting the recursive structure. Operators act as discrete manifestations of the substrate grammar.

Remark 1. *Operators may preserve, repair, or collapse recursive evolution. In practice, operators are implemented as "repair rules" (rounding, thresholding, damping) or as projections onto structured lattices (Gaussian, Eisenstein, cyclotomic).*

2 Mapping Table

3 Visualization

Figure 1 illustrates the correspondence between UNNS operators and substrate actions. Collapse operators pull values inward; Inlaying operators snap to lattice embeddings; Adoption extends outward; Evaluation fixes symbolic states.

4 Conclusion

This mapping clarifies how low-level repair operators used in computational explorers correspond to the higher-level grammar of the UNNS substrate. By making the theoretical roles explicit, the bridge between interactive demos and formal recursive theory is established.

Implemented Operator	**UNNS Grammar**	**Conceptual Role**
`threshold_zero`	Collapse	Absorbs small echoes into substrate (zero)
`merge_close`	Collapse/Normalize	Forces near-equal states to converge
`gaussian_round`	Inlaying ($\mathbb{Z}[i]$)	Snap recursion to Gaussian lattice
`eisenstein_round`	Inlaying ($\mathbb{Z}[\omega]$)	Snap recursion to hexagonal lattice
`cyclotomic_proj(`p`)`	Inletting/Inlaying	Embed into pth root of unity lattice
`roundInt / roundFixed`	Evaluate	Snap recursion to discrete numeric states
`damp(`α`)`	Normalize	Dissipates recursive energy, stabilizes growth
`adopt_shift(`k`)`*	Adoption (proposed)	Imports new states from deeper nests

Table 1: Mapping of code-level operators to UNNS substrate grammar. (*) Adoption operator not yet implemented.

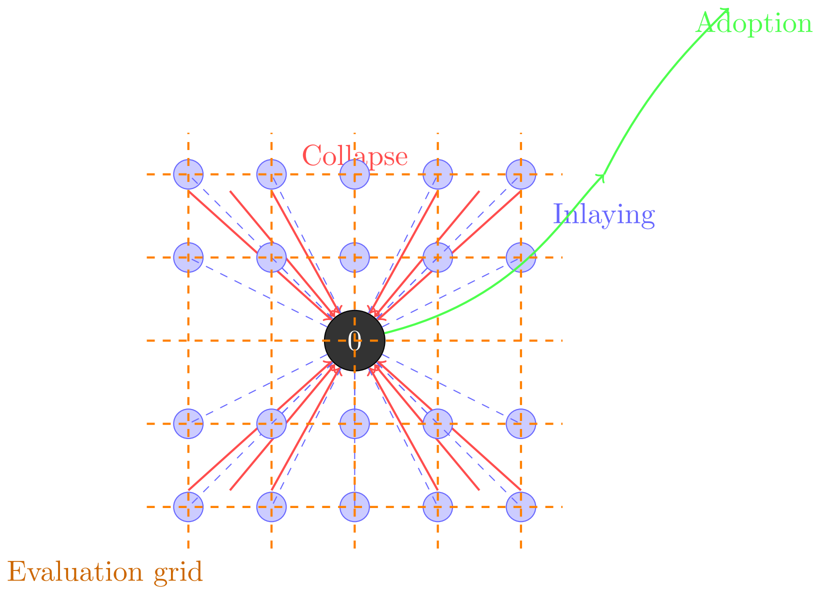

Figure 1: Schematic illustration of UNNS operators: Collapse (red), Inlaying (blue), Adoption (green), and Evaluation grid (orange).