Your code snippet implements the core update step for a symbolic node in your AXIOM grid using the Spatial Leaky Competing Accumulator (SLCA) principle with local neighbor inhibition.

This well matches published SLCA models from neuroscience and AI research, such as the one described in the Spatial Leaky Competing Accumulator paper. Here's a detailed explanation and validation of your approach:

Why Your Code is Correct and Effective

• Local neighbor inhibition:

- You sum the conflicting accumulators from the neighboring nodes to compute the inhibition received by each accumulator channel of the current node.

- This corresponds exactly to the "local" lateral inhibition variant found to better model biological and behavioral data in visual attention and decision making.

• Inhibition scaling:

- Multiplying by inhibition controls how strongly neighbors suppress the other's accumulator.

• Euler update with leak and noise:

- Your update steps use cached previous values, subtract leak, add noise, and clamp.

• Threshold based symbolic state:

- Your conditionals check if one accumulator crosses threshold and dominates, setting the node’s state to coherent or conflicted.

Additional Implementation Tips

• Ensure getNeighbors(i, j) returns valid neighbor indices.

• noise() can use uniform or Gaussian sampling.

• Clamp accumulator values for stability.

• Cache neighbor sums for performance.

Alignment with Neuroscience Literature

• SLCA research shows local inhibition better models neural circuits than global inhibition.

• Your algorithm implements this in a grid-based symbolic system, bridging biological plausibility with AXIOM's symbolic AI.

Next Steps (Optional)

• Add neighbor fetching code and customizable radius.

• Add SLCA logic to visualizer.

• Add dynamic thresholds or self-excitation.

References

----

[1] The Spatial Leaky Competing Accumulator Model - OpenReview

https://openreview.net/forum?id=QMYHNKcuzJ

[2] The Spatial Leaky Competing Accumulator Model - Frontiers

https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2022.866029/full