Principles for Engineered Emergence

Jacob Beal
MIT CSAIL

“Engineered Emergence”

Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.

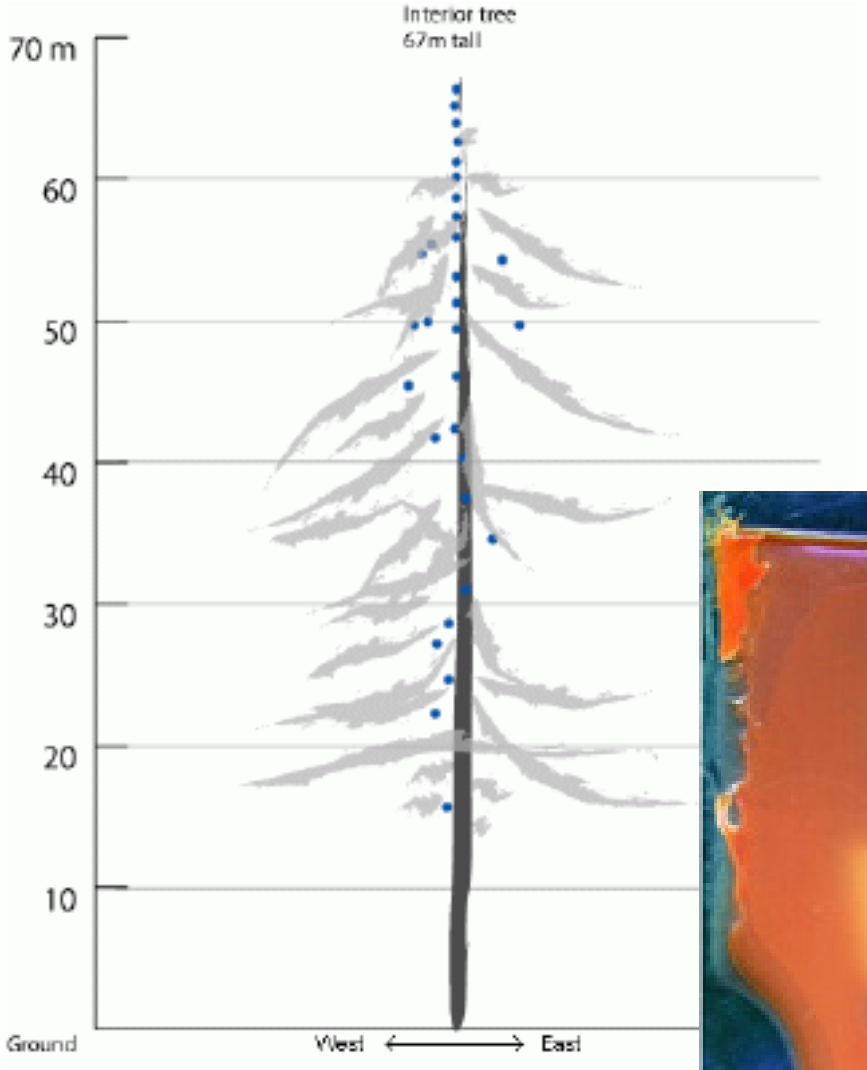
How do I clean up the mess?

Four Useful Principles

- Self-Scaling
- Sparseness
- Gradual degradation
- Failure simplification

OK, but how hard is it to apply them?

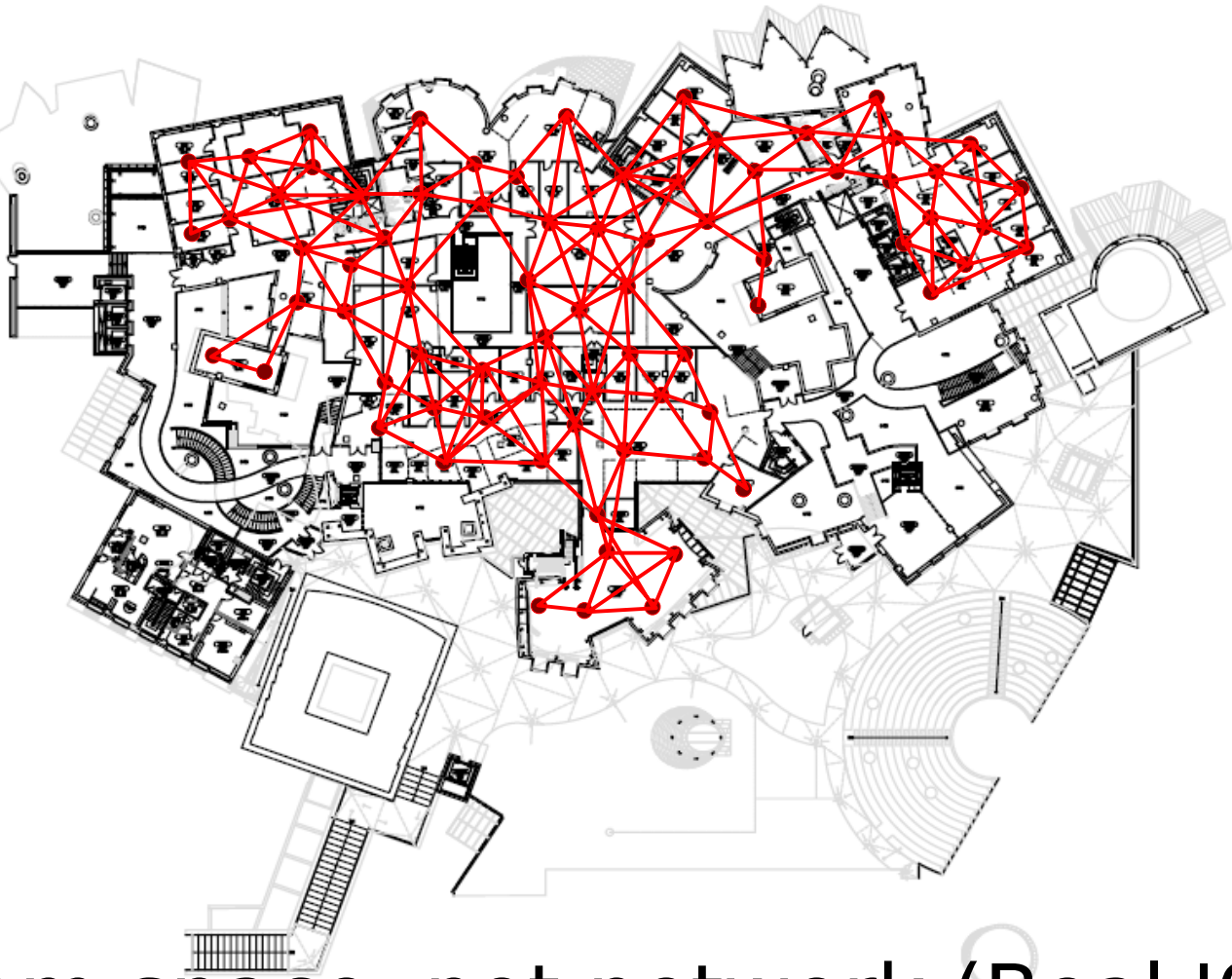
Spatial Computing



Amorphous Medium

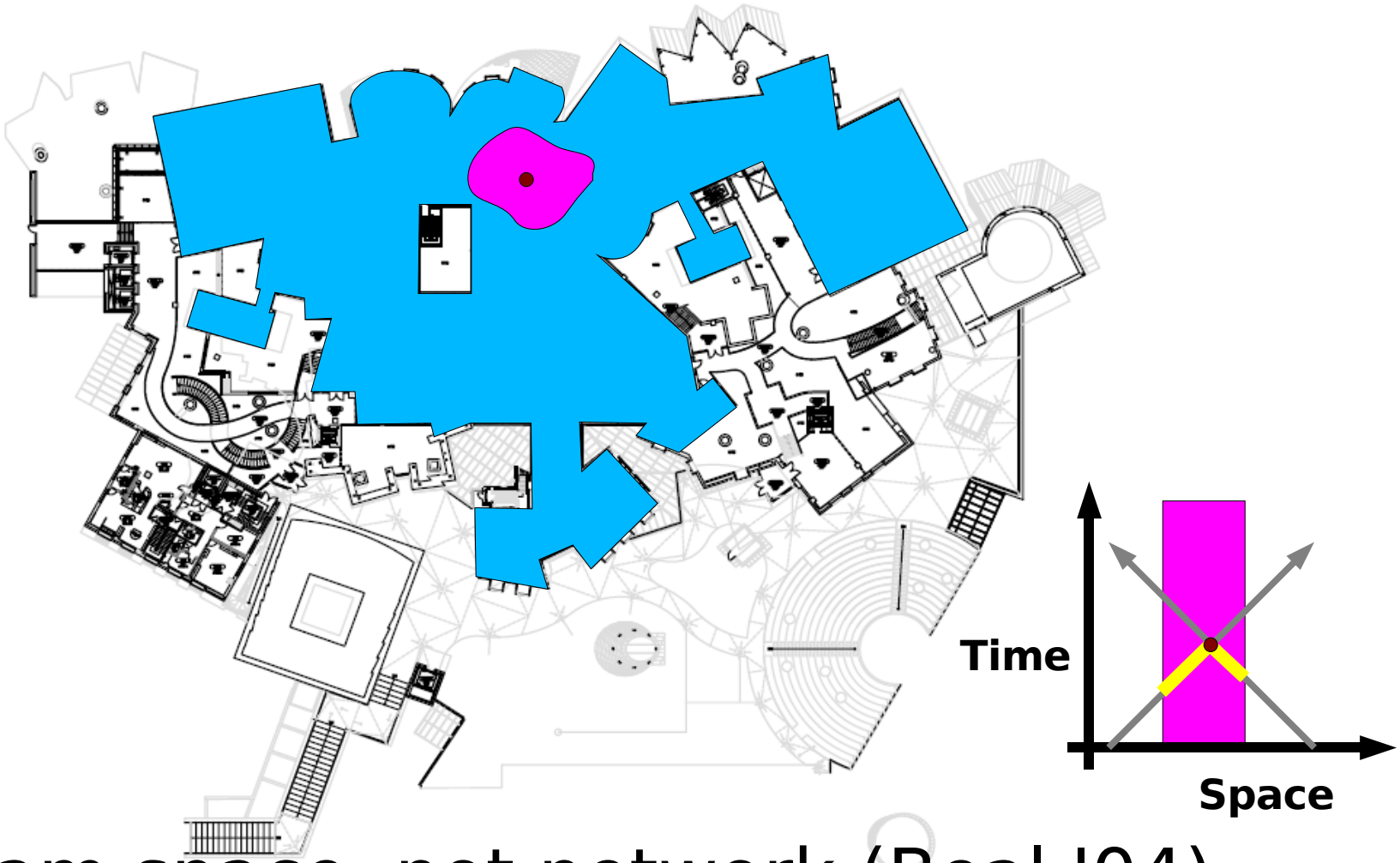
Program space, not network (Beal '04)

Amorphous Medium



Program space, not network (Beal '04)

Amorphous Medium



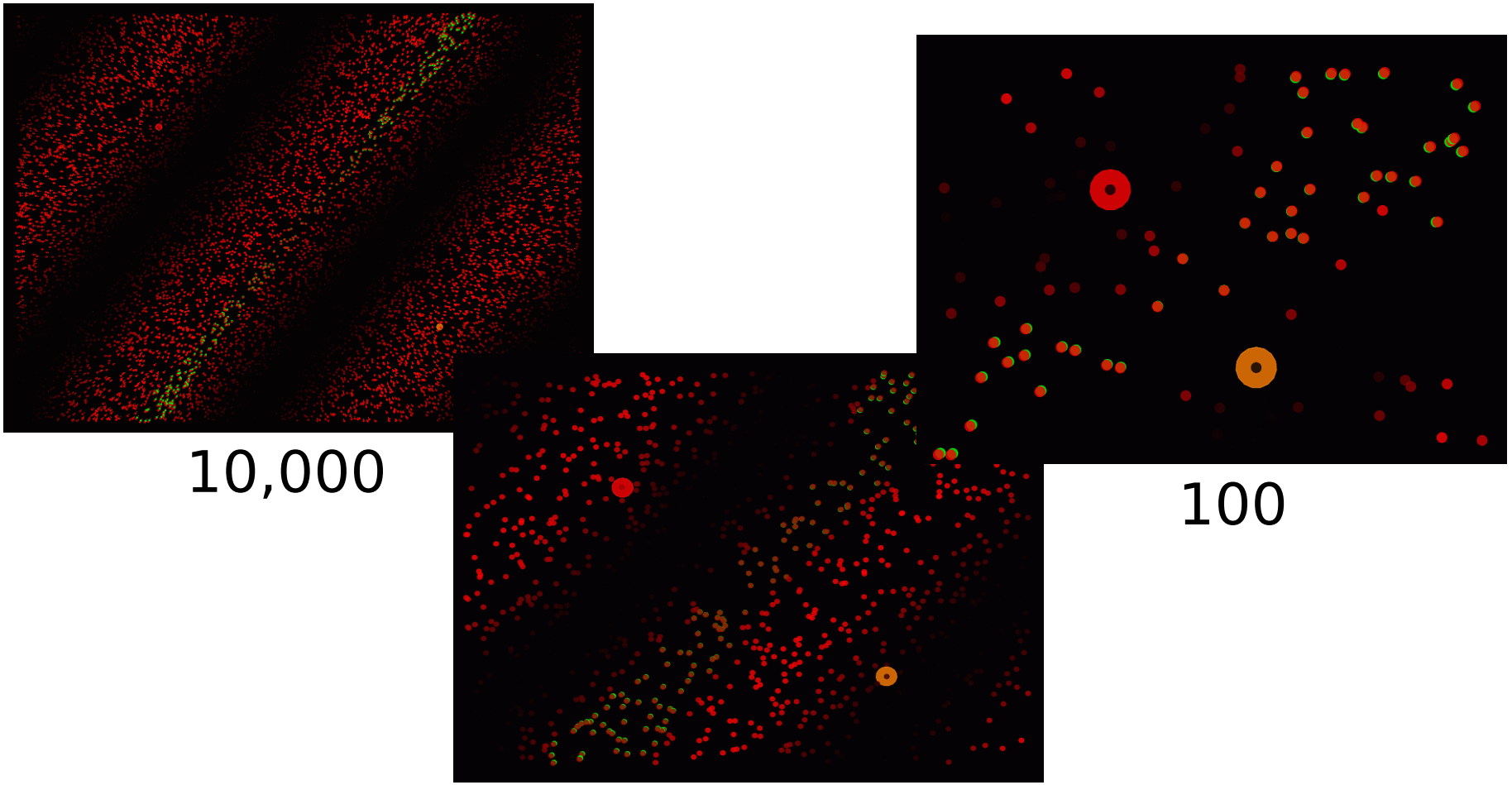
Program space, not network (Beal '04)

Failure Simplification: Aggregate Values

Information is lost, but failures change summaries, rather than individuals.

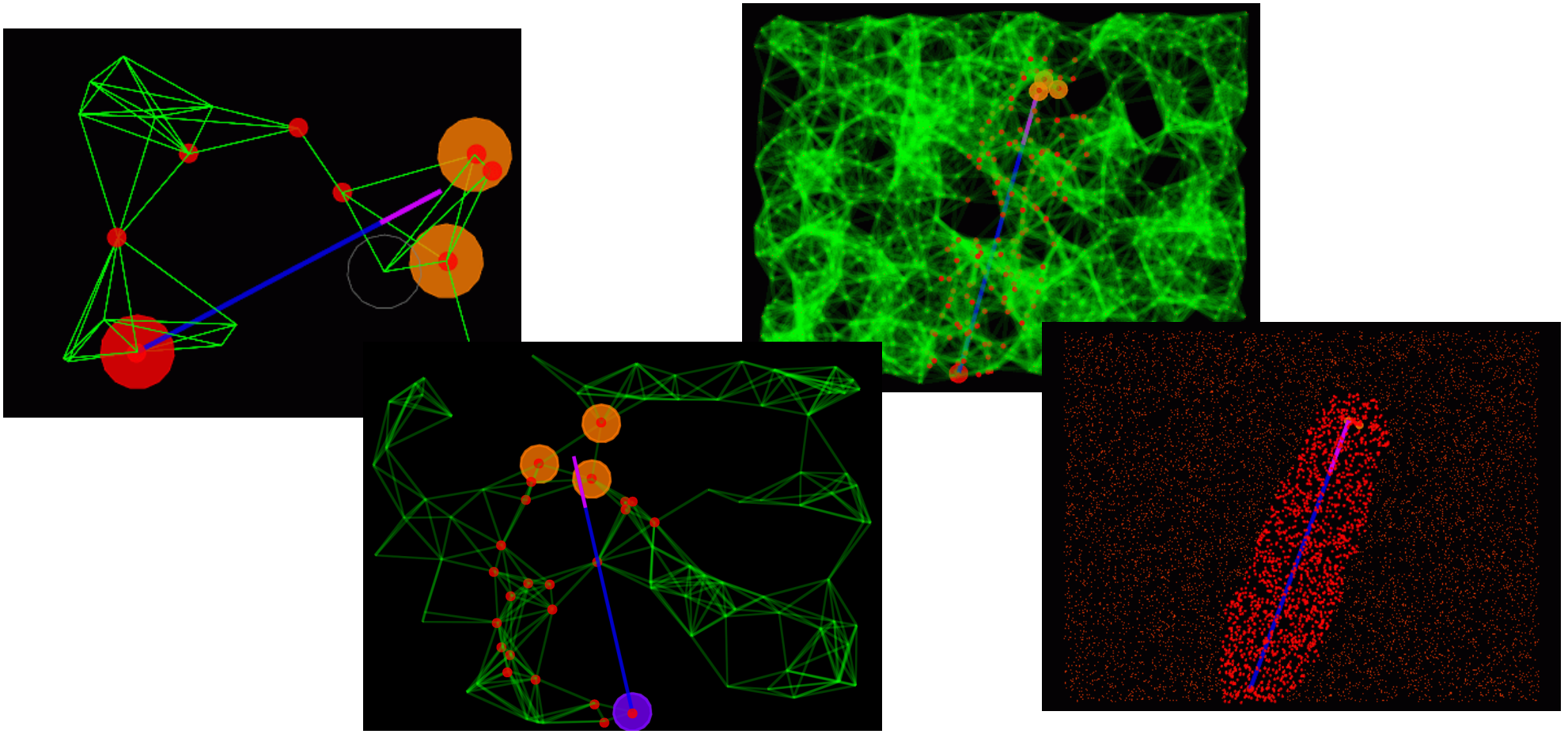
Gradual Degradation: Implementation Details

- Plane wave at different resolutions:



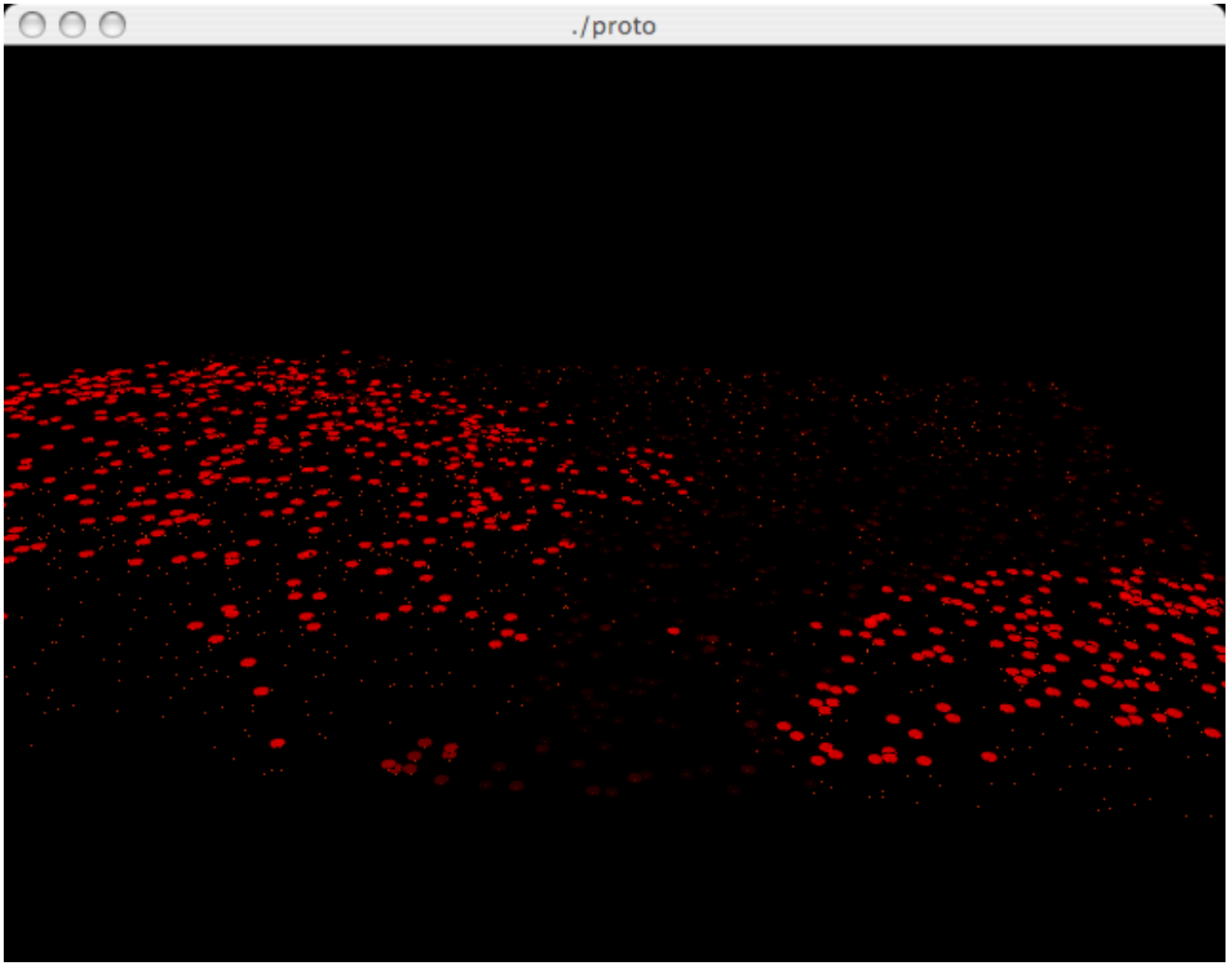
Self-Scaling: Neighborhood Ops

- Proto (Beal & Bachrach '06):
  - Scales by increasing resolution



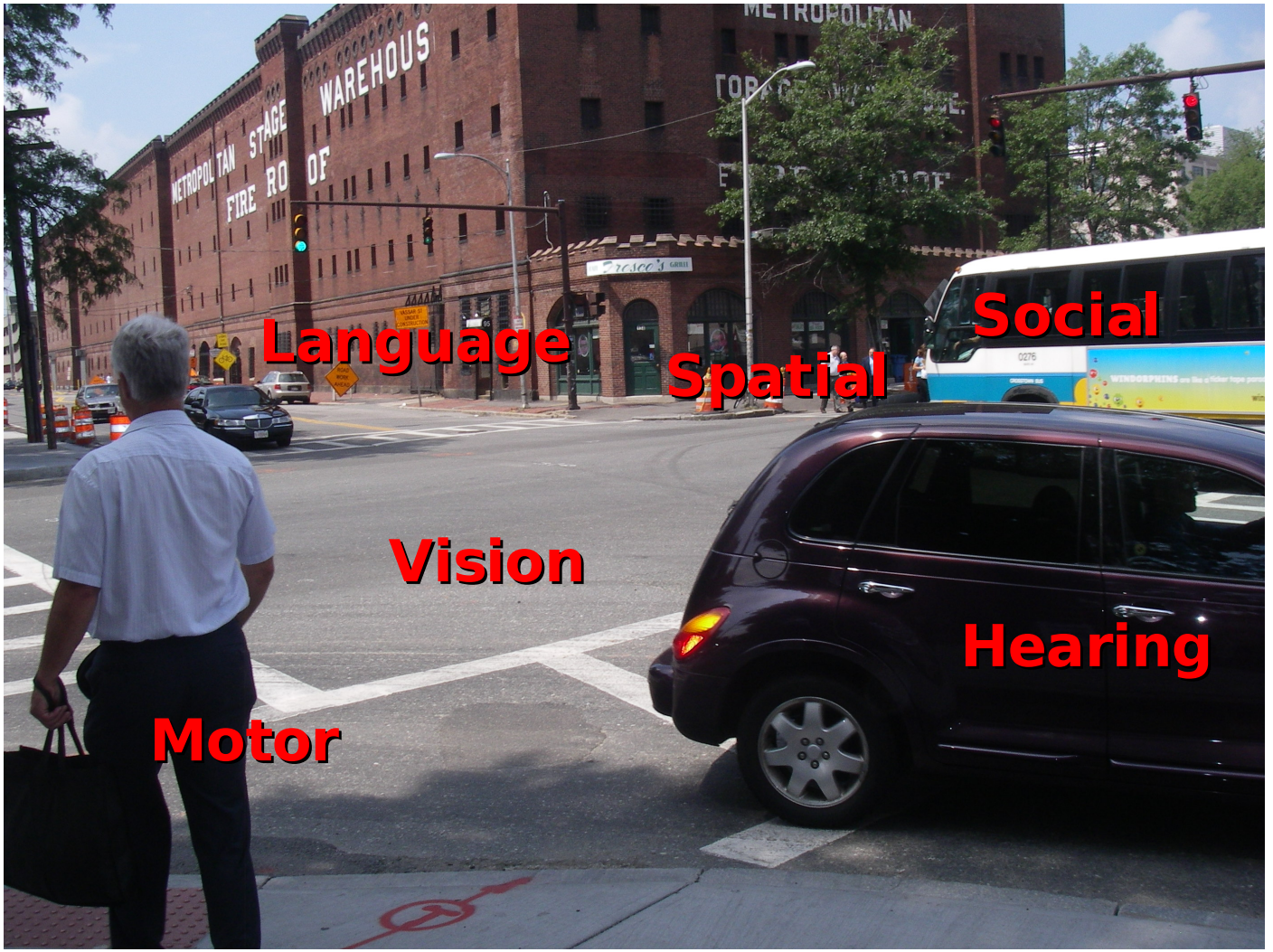
Sparseness: Symmetry Breaking

- Temporary leadership via 1/f noise

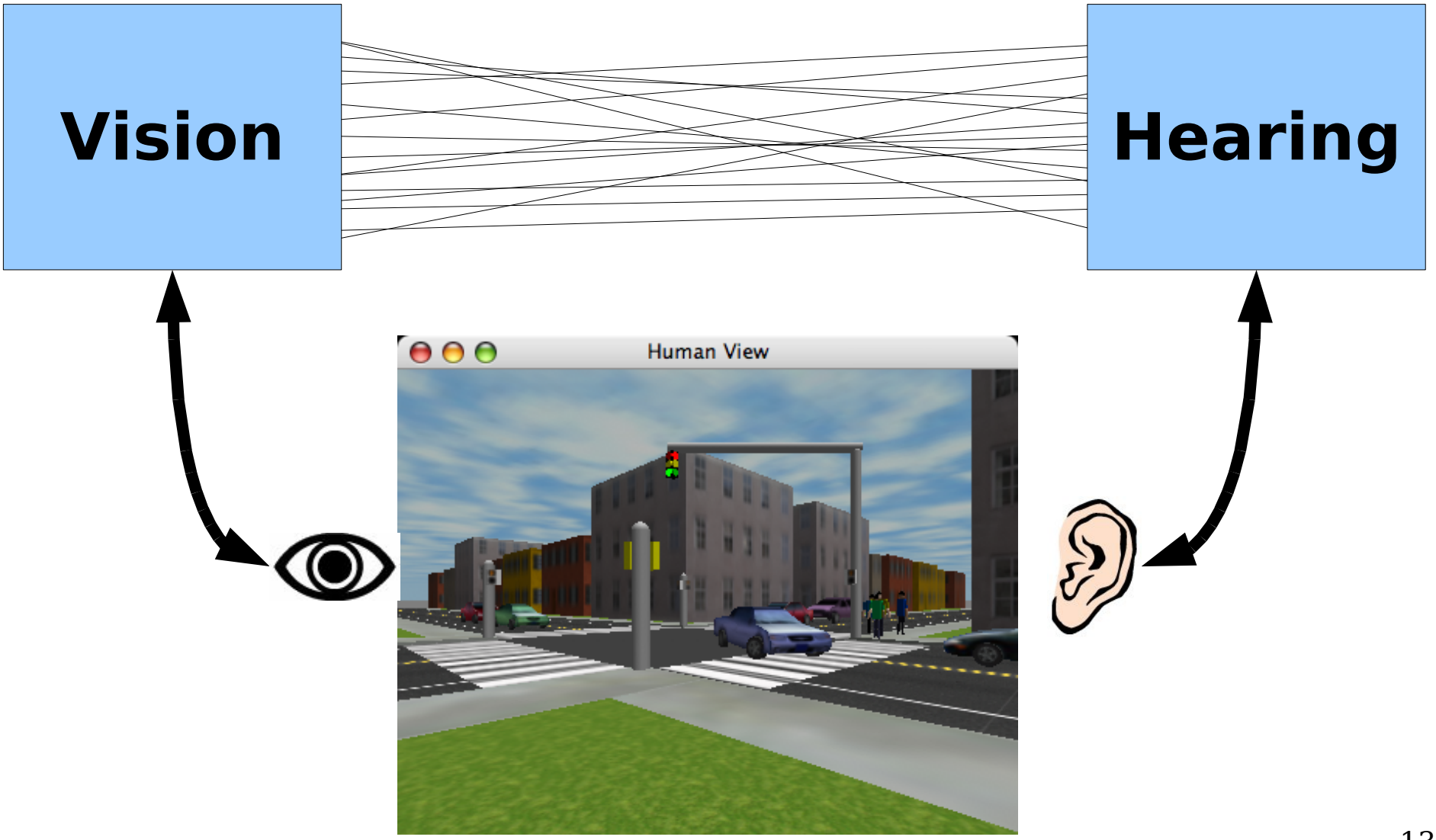


Cognitive Models

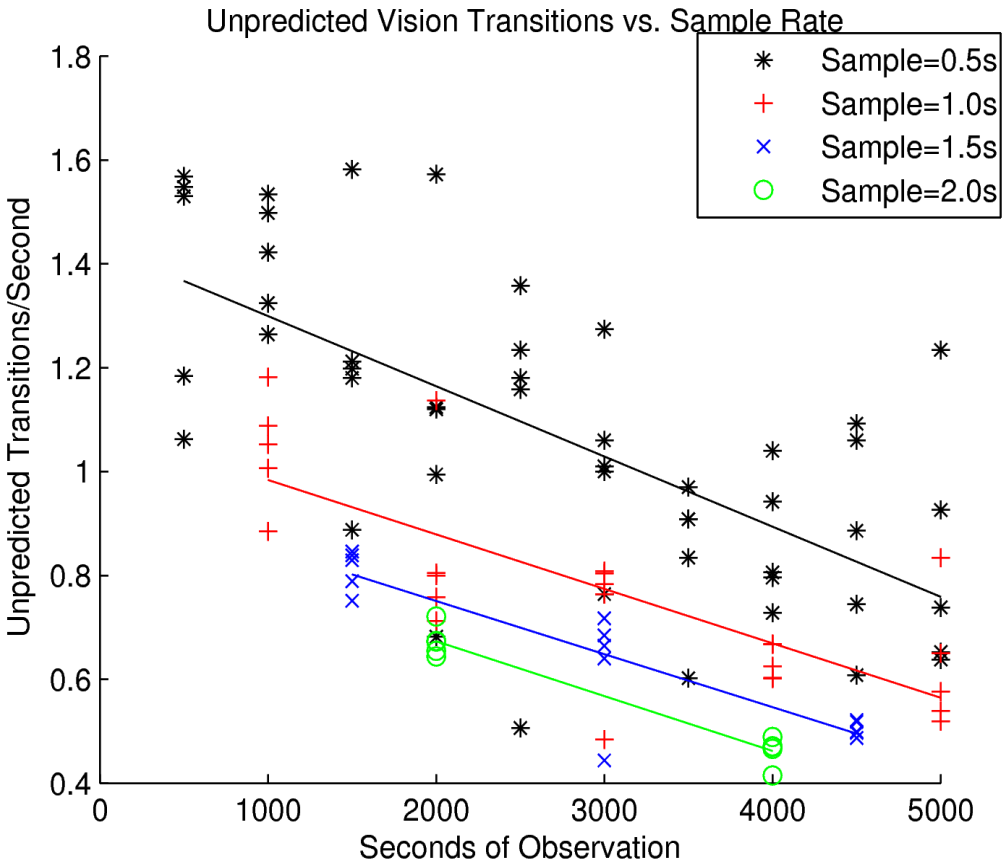
How might specialist parts learn to work together as a unified mind?

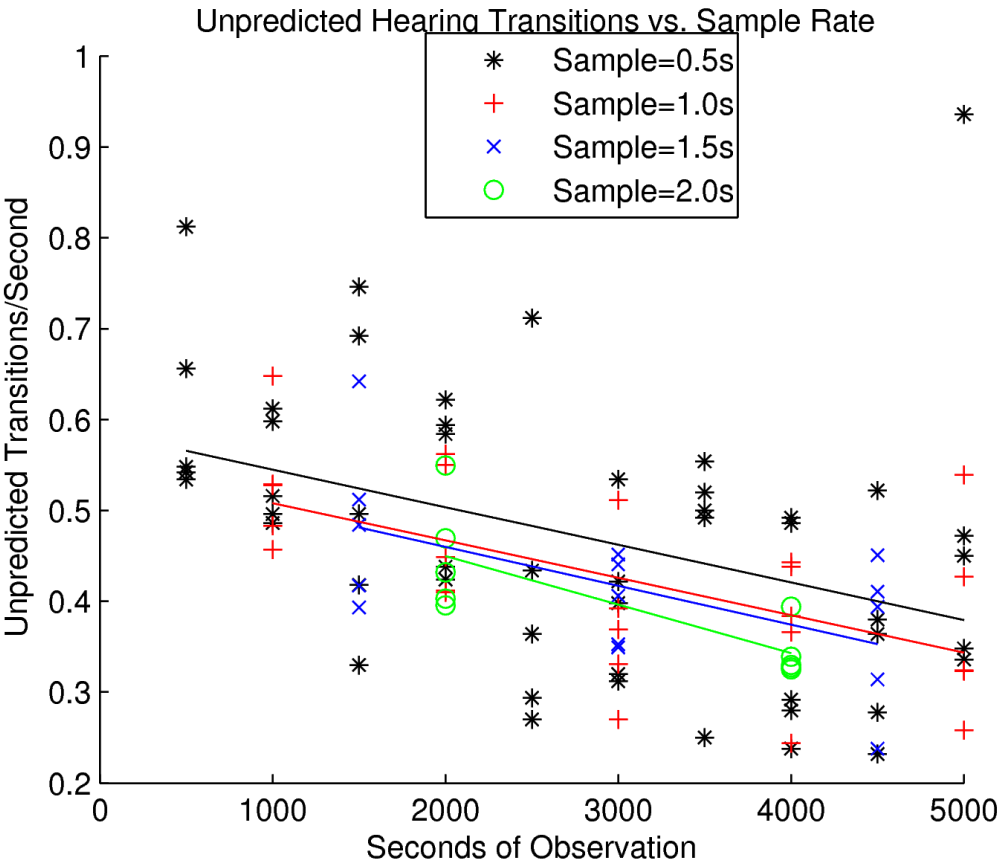


Learning by Learning to Communicate

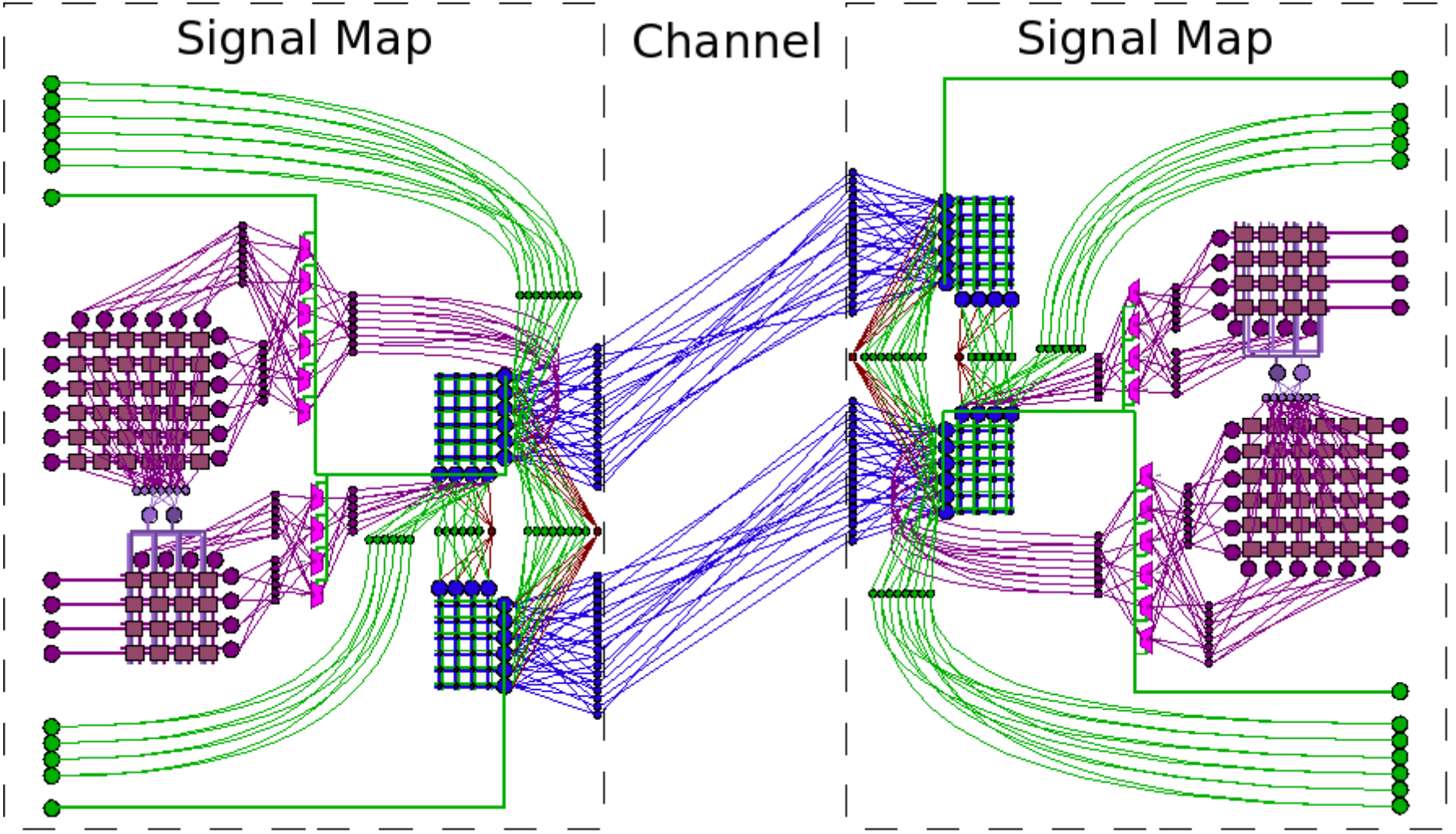


Vocabulary agreement improves sensory prediction



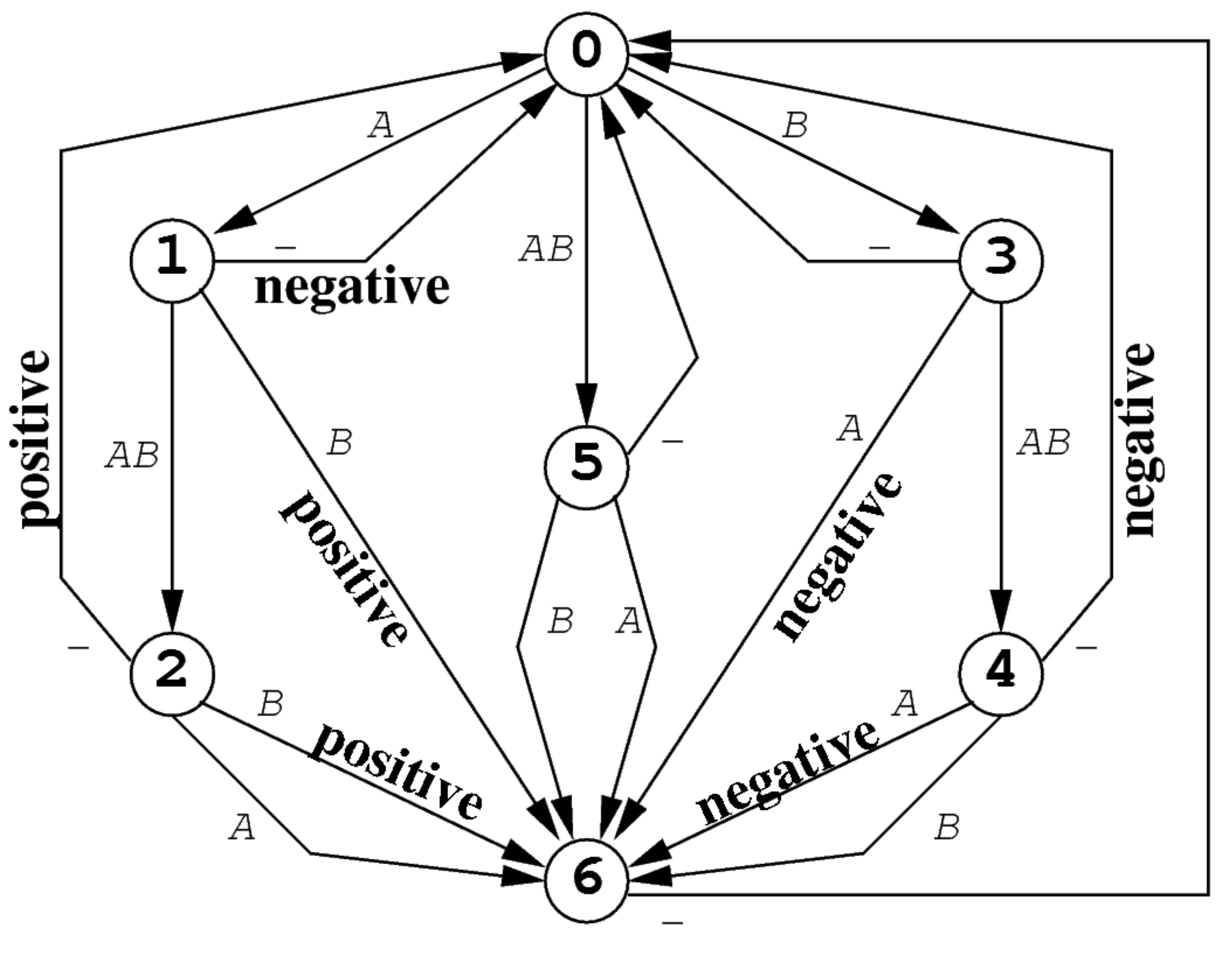


Sparseness: Self-Organized Symbolic Communication



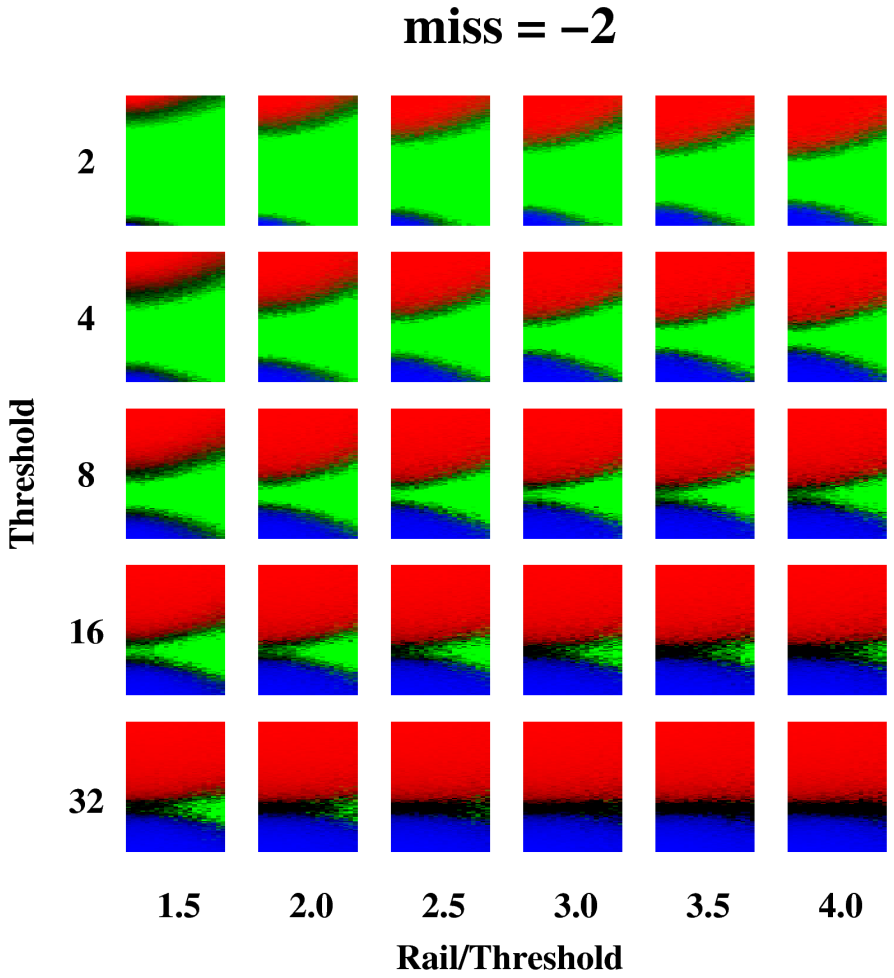
Self-Scaling: IIES

- Incremental capture of quasi-independent examples from highly correlated input

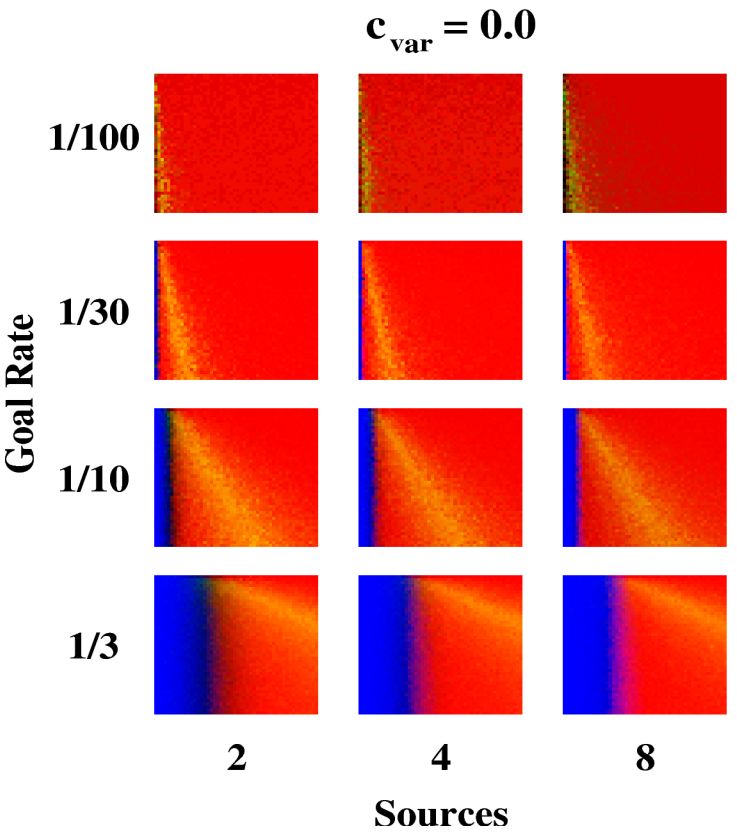


Gradual Degradation: Dossiers

Coincidence Detector:

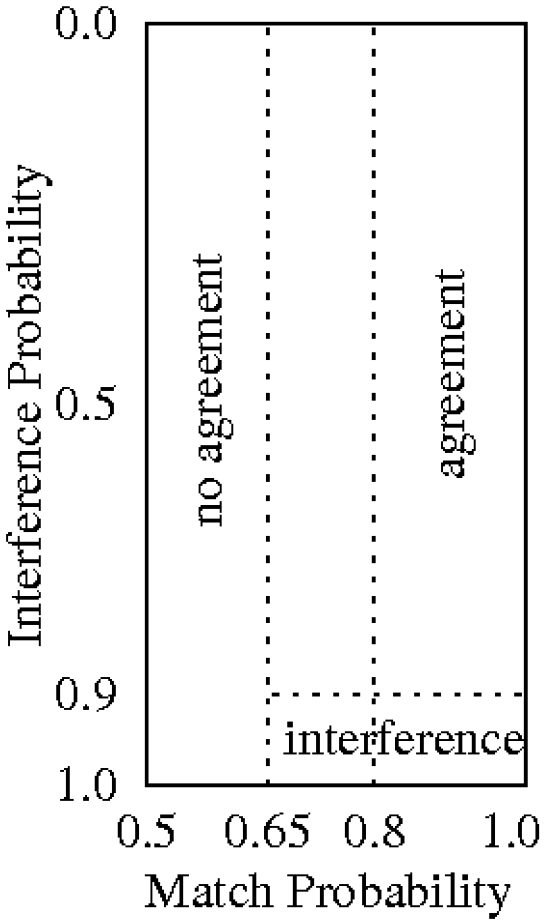


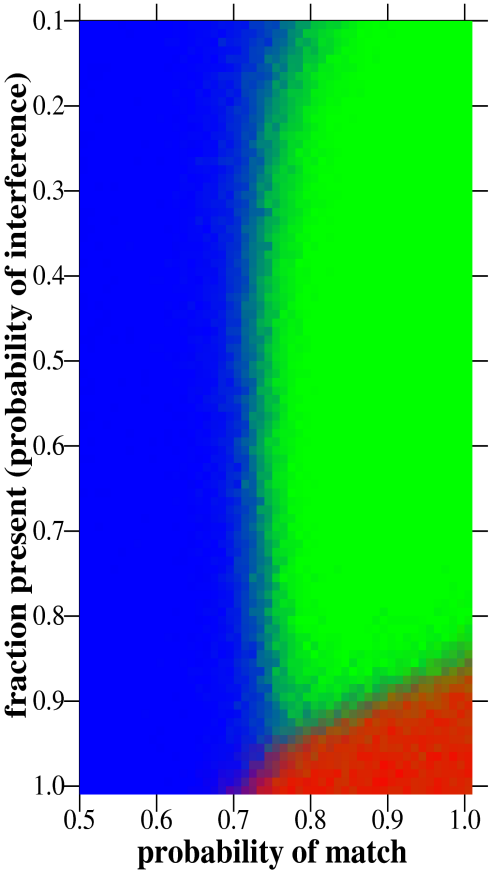
Event Throttle:



Failure Simplification: Pre-emptive Failure

- Coincidence Detector: if it's not a fast success, it's a failure.





Contributions

- Four tools for simplifying ugly designs
  - self-scaling, sparseness, gradual degradation, failure simplification
- Examples of use in two domains

Where else can we apply them?

Can we analyze them formally?

What other tools can we discover?