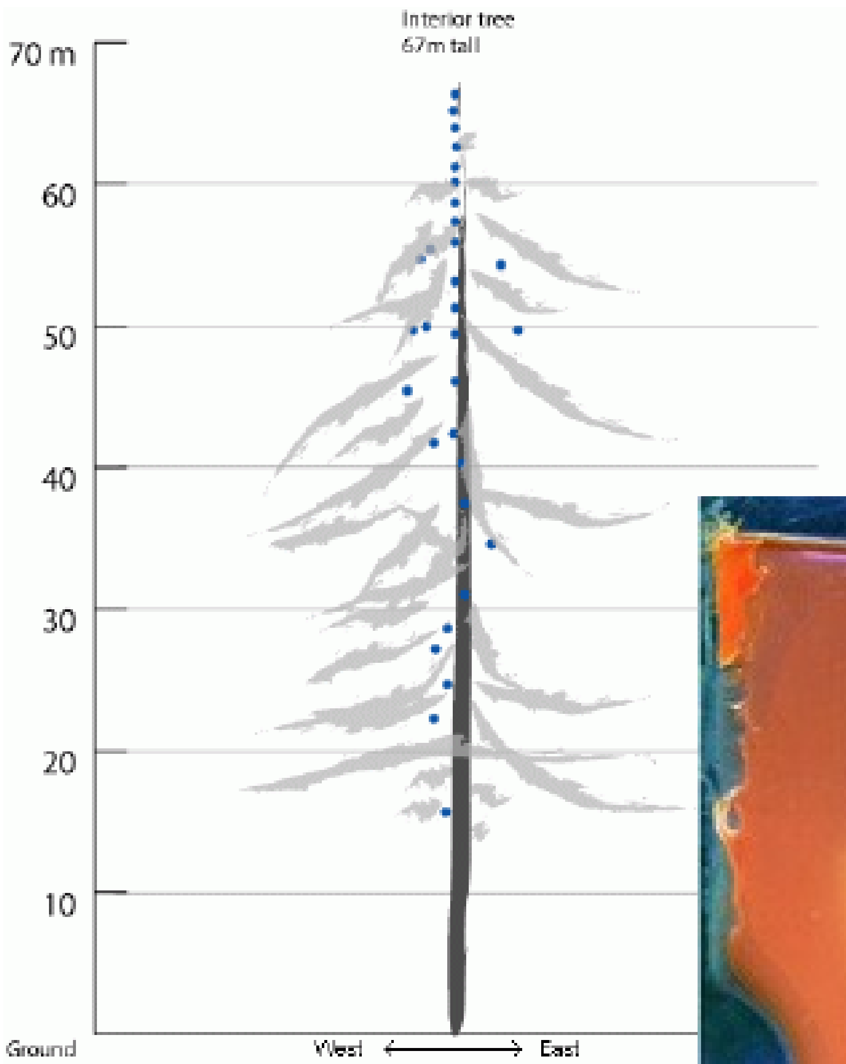
Principles for Engineered Emergence

Jacob Beal
MIT CSAIL

Spatial Computing



Cognitive Architectures

- How do the parts learn to work together?
  - Vision
  - Language
  - Motor
  - Social

“Please pass the coffee”

“Engineered Emergence”

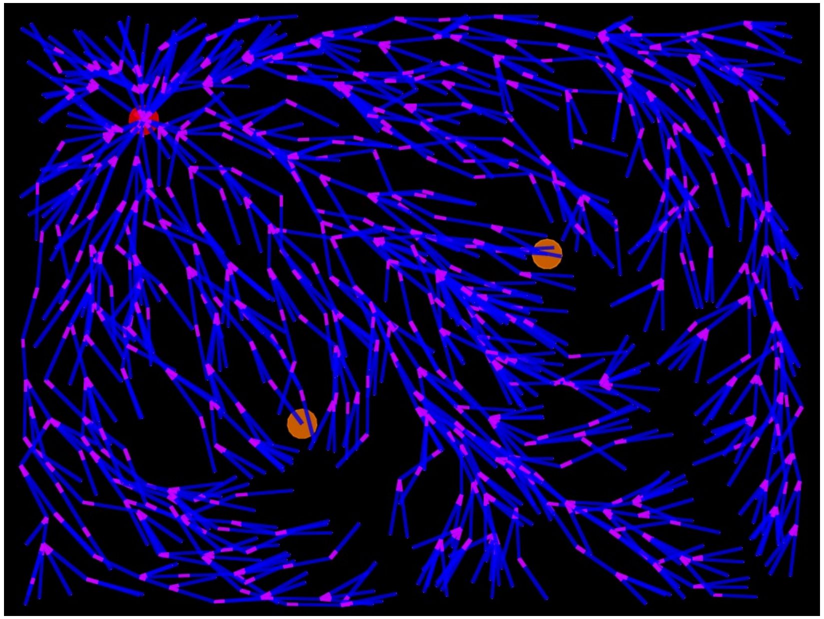
Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.

- Programming language approach
  - Primitives
  - Means of composition
  - Means of abstraction

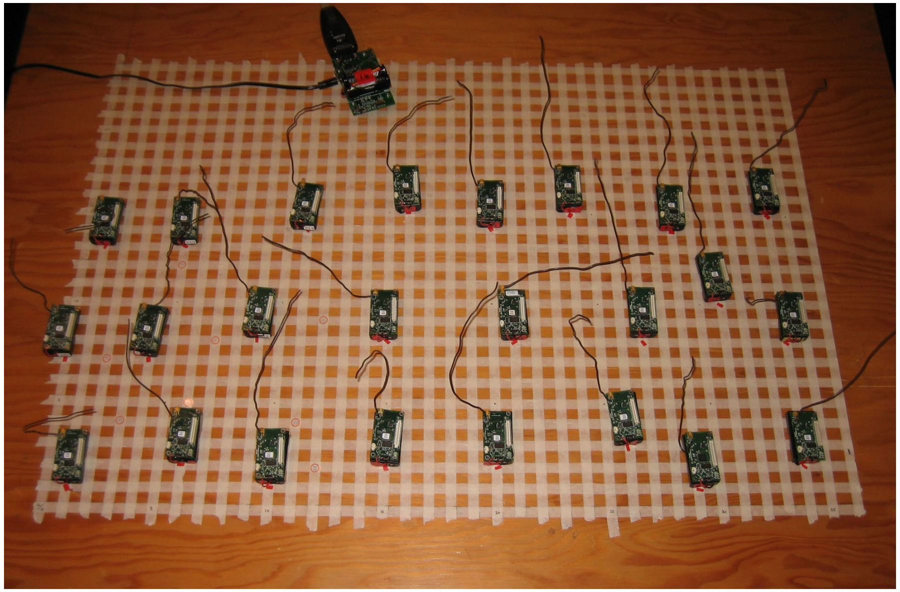
How do I clean up the mess?

Spatial Computers

- Devices spread through space whose ability to interact depends on geometry



Simulation

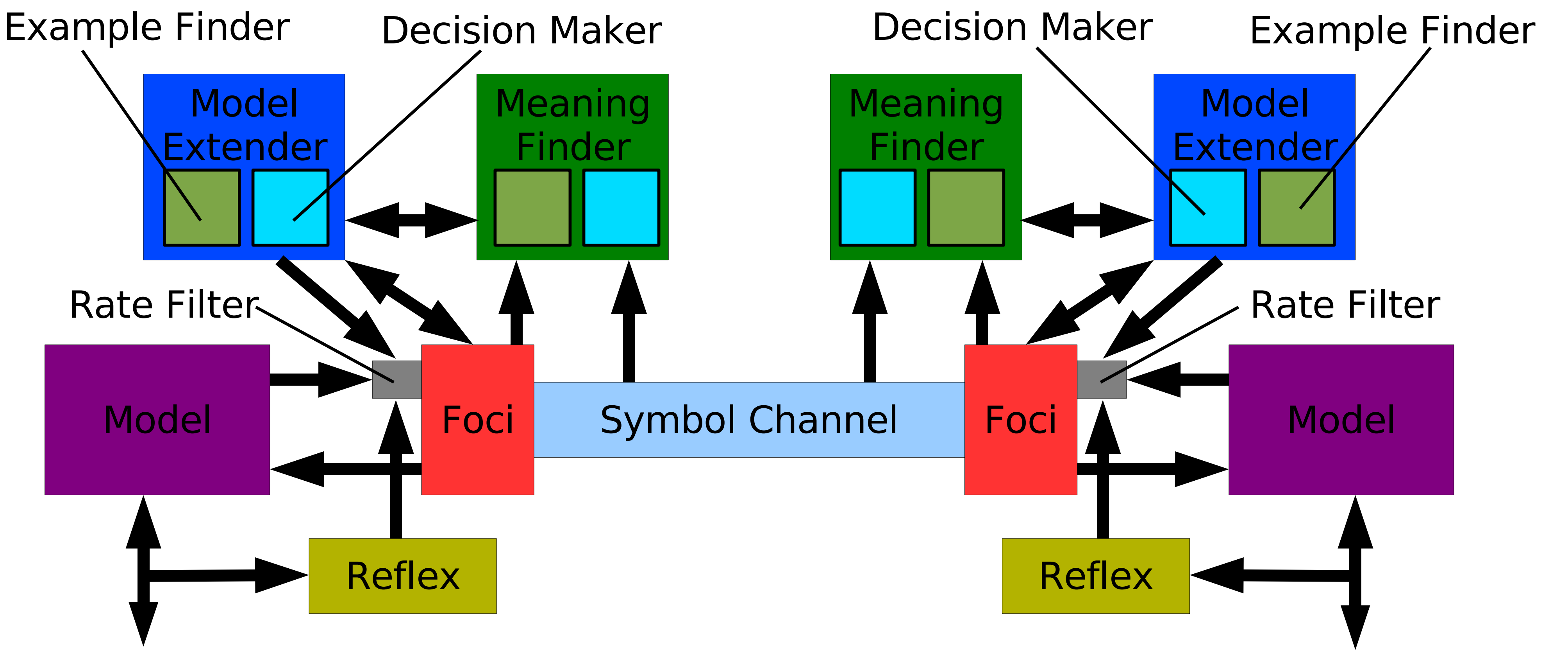


Mica2 Motes

(Bachrach & Beal '06)

Cognitive Architectures

- How can we build **towards** human-like competence and flexibility?
  - Example: communication bootstrapping



Four Useful Principles

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

OK, but how hard is it to apply them?

Self-Scaling

- Use when you don't know the relationship between the behavior you want and the details of its implementation
- Decoupling through geometry:
  - specification of behavior (units)
  - implementation details (coordinate system)

Sparseness

- Use when device need to make non-interfering decisions independently.
- Decoupling by making unwanted interactions rare.

If at first you don't succeed, just try again.

Gradual Degradation

- Use when you don't understand or can't control the environment.
- Decoupling by low sensitivity to
  - Implementation details
  - Parameter values
  - Conditions of execution

Failure Simplification

- Use when you don't understand or can't prevent failures
- Decouple by preferentially selecting failure type

We're used to preventing failures.

What if we just manage their impact?

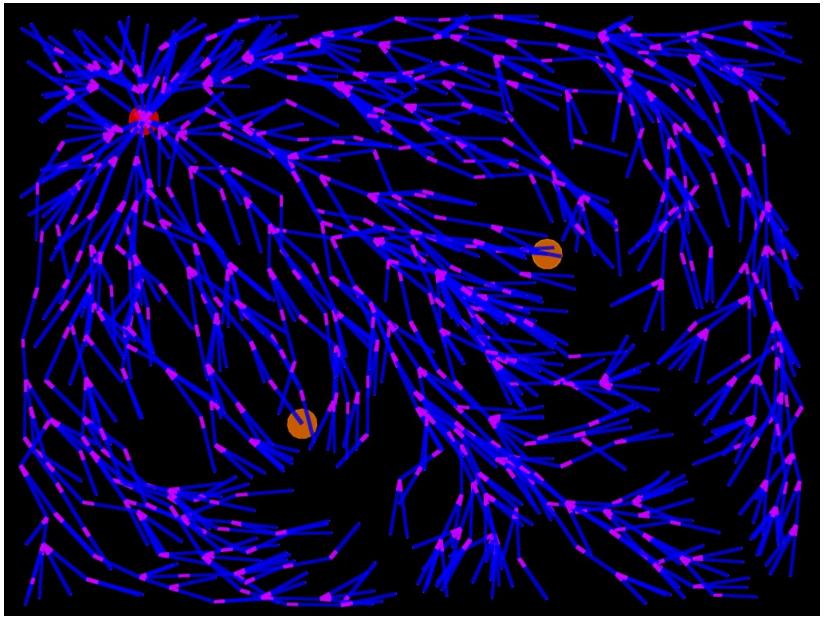
Failure Simplification

- These people are executing an apparently impossible distributed simulation algorithm.

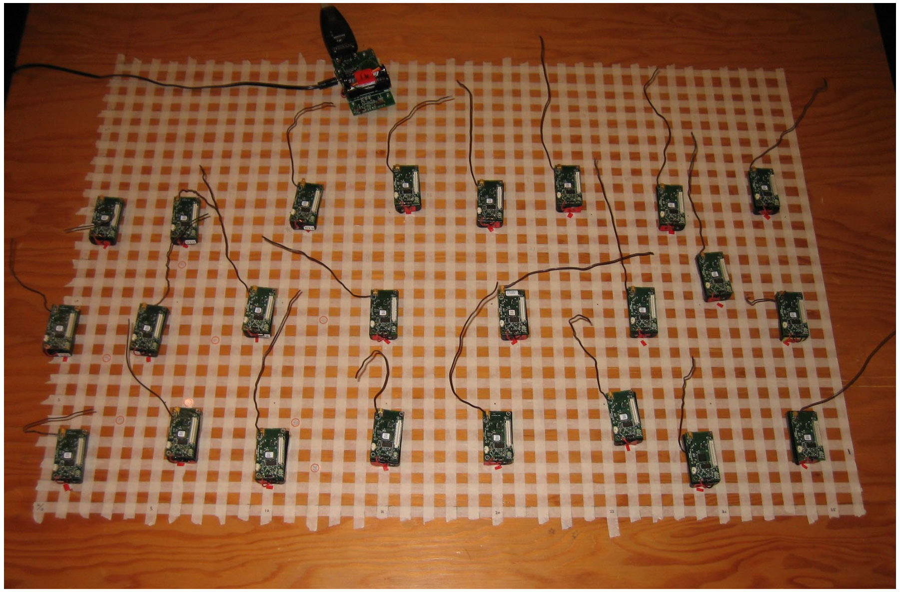
Not all failures are important!

Spatial Computers

- Devices spread through space whose ability to interact depends on geometry



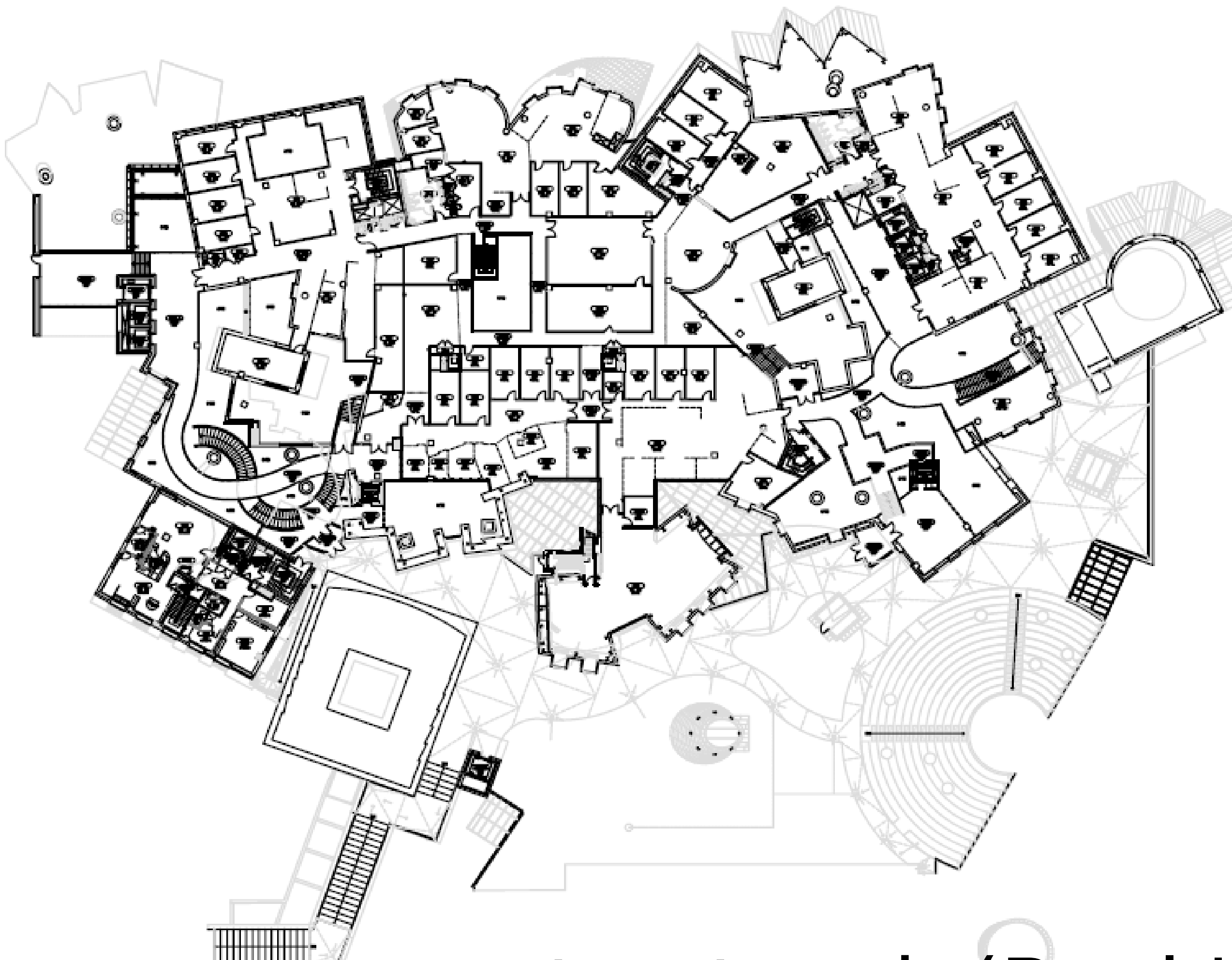
Simulation



Mica2 Motes

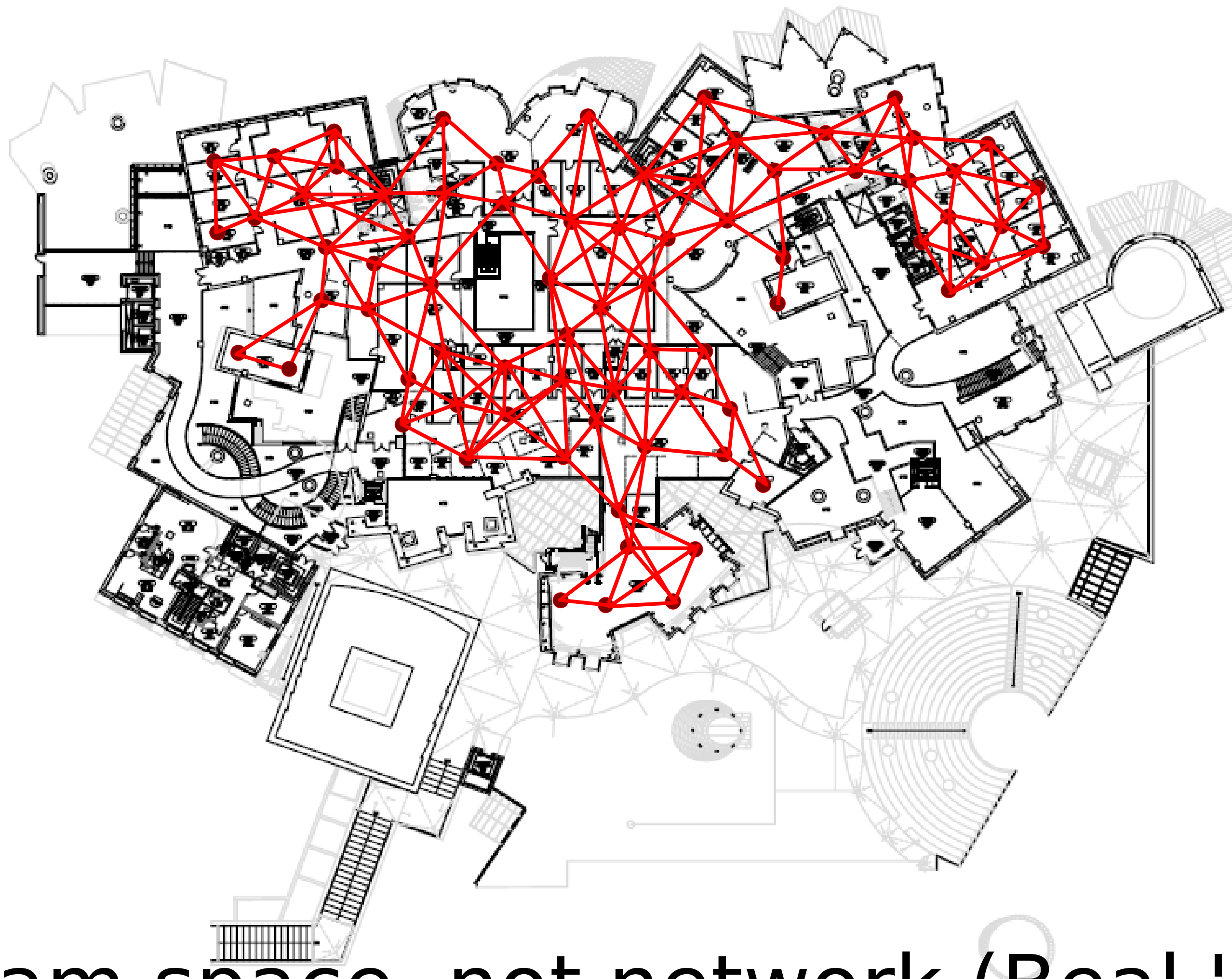
(Bachrach & Beal '06)

Amorphous Medium



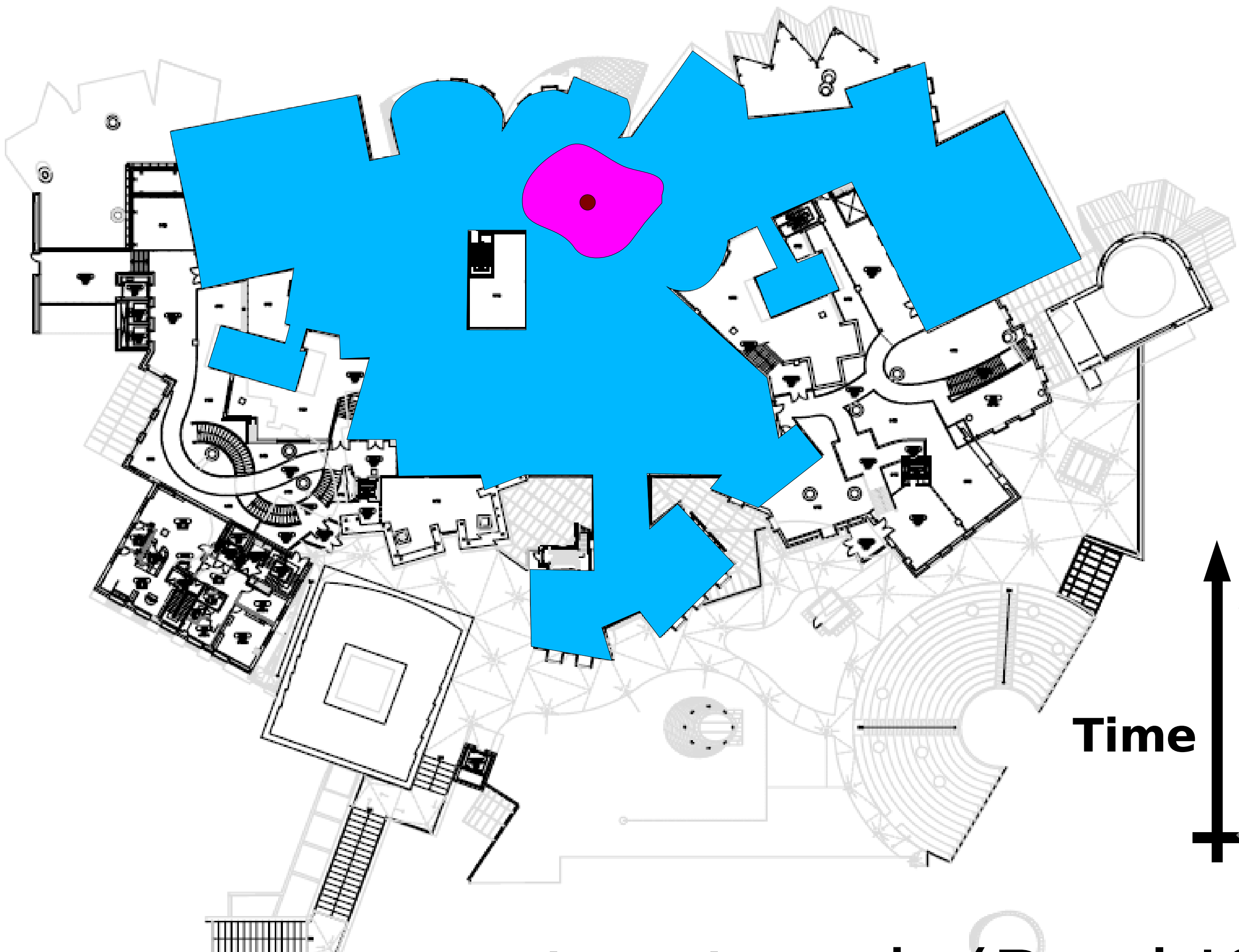
Program space, not network (Beal '04)

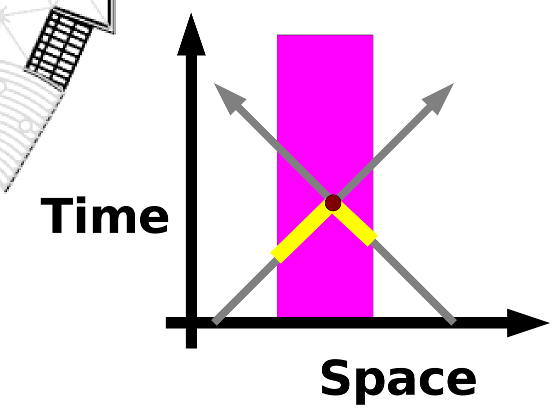
Amorphous Medium



Program space, not network (Beal '04)

Amorphous Medium





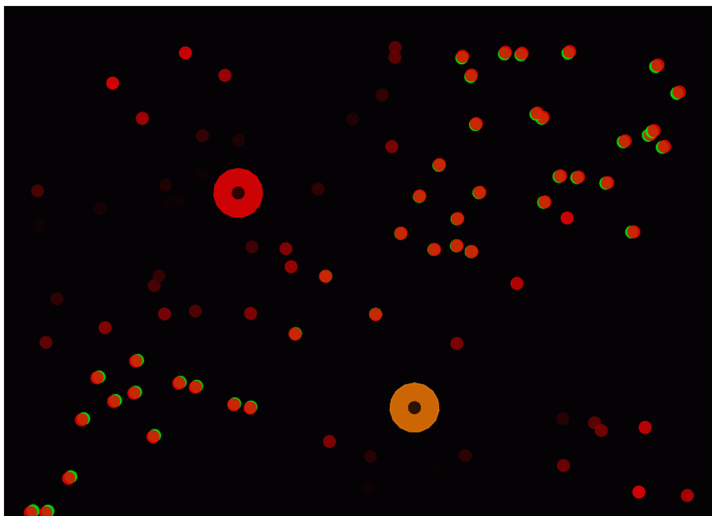
Program space, not network (Beal '04)

Fail. Simp.: Aggregate Values

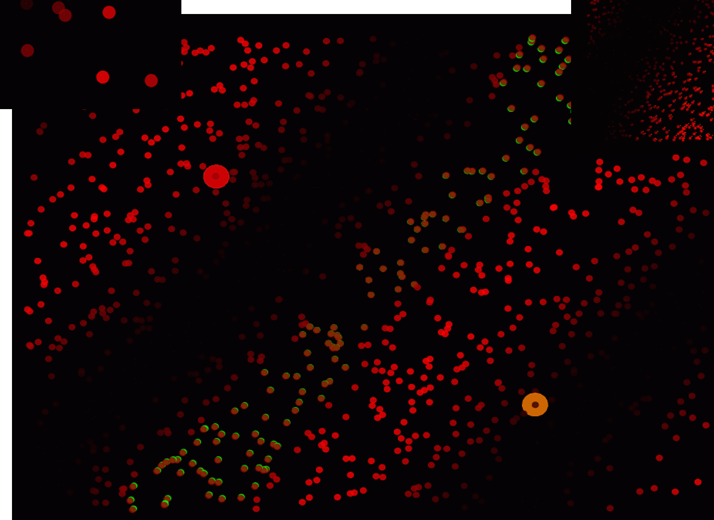
Information is lost, but failures change summaries, rather than individuals.

Gradual Degradation: Implementation Details

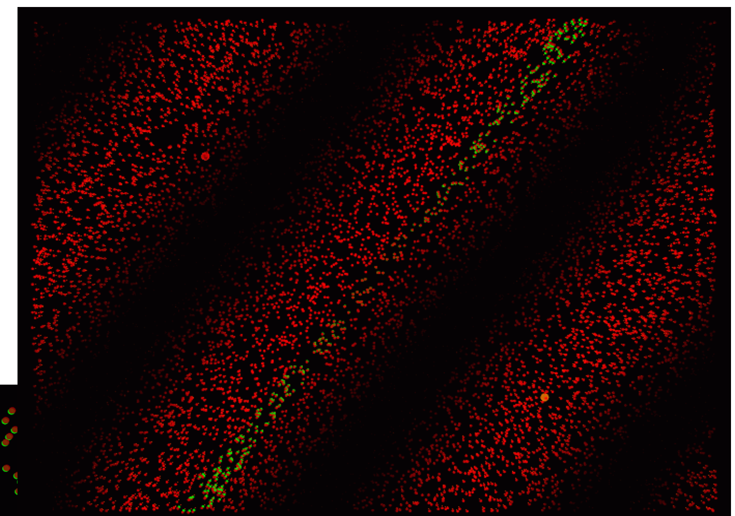
- Plane wave at different resolutions:



100



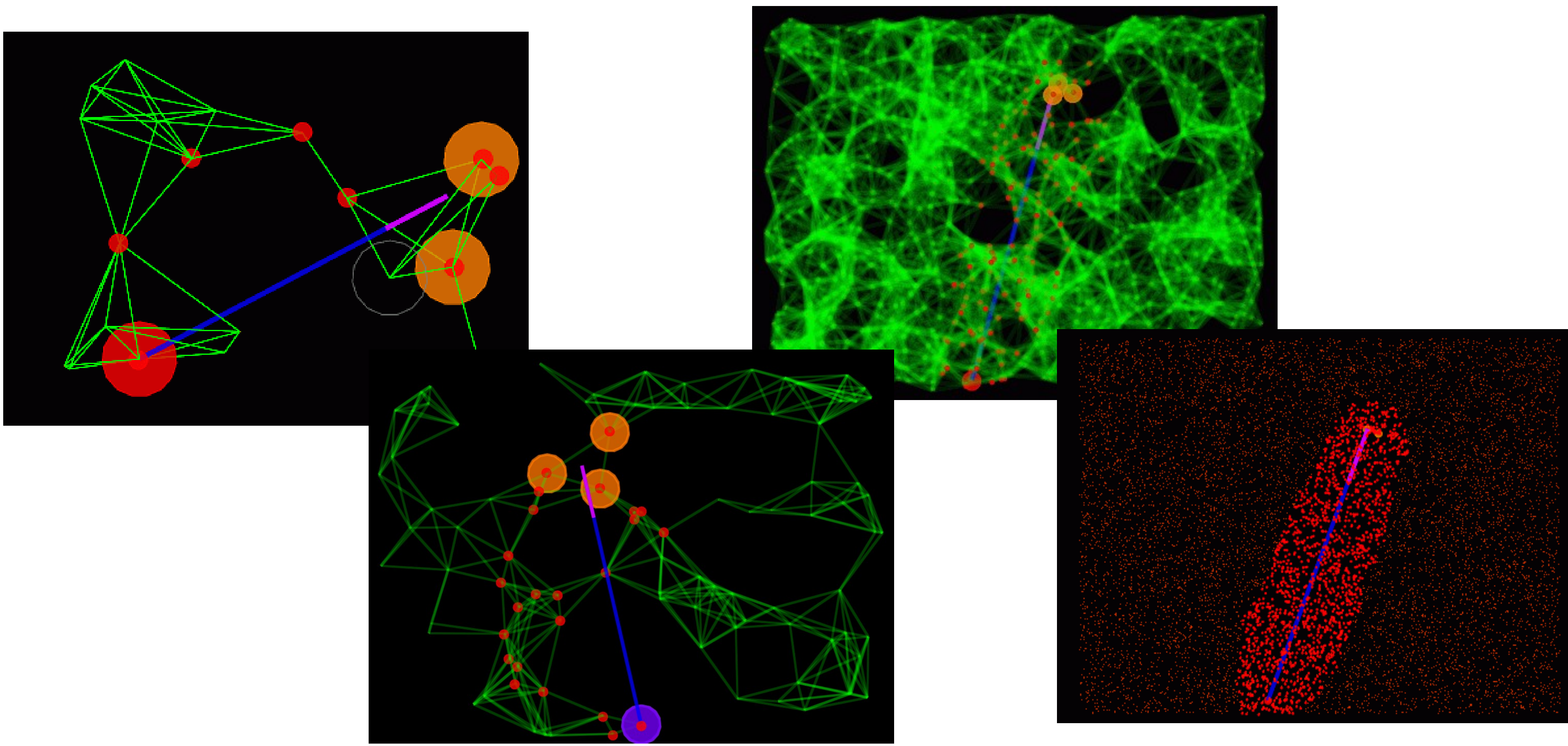
1,000



10,000

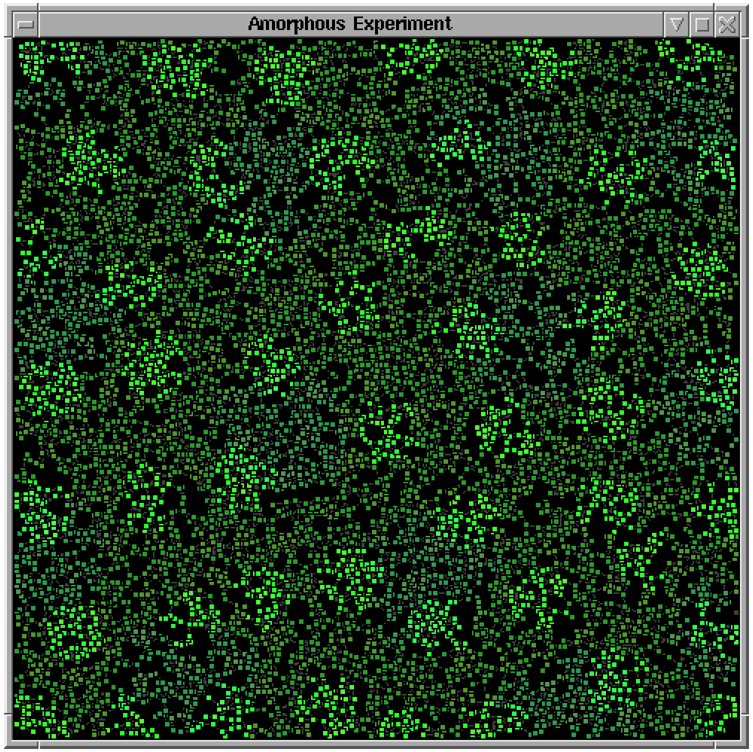
Self-Scaling: Neighborhood Ops

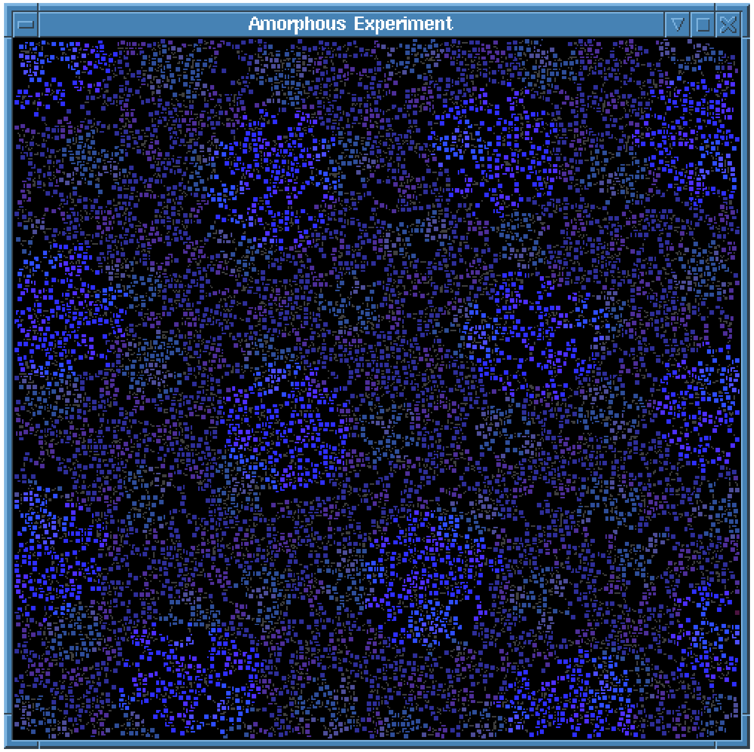
- Proto (Beal & Bachrach '06):
  - Scales by increasing resolution



Sparseness: Symmetry Breaking

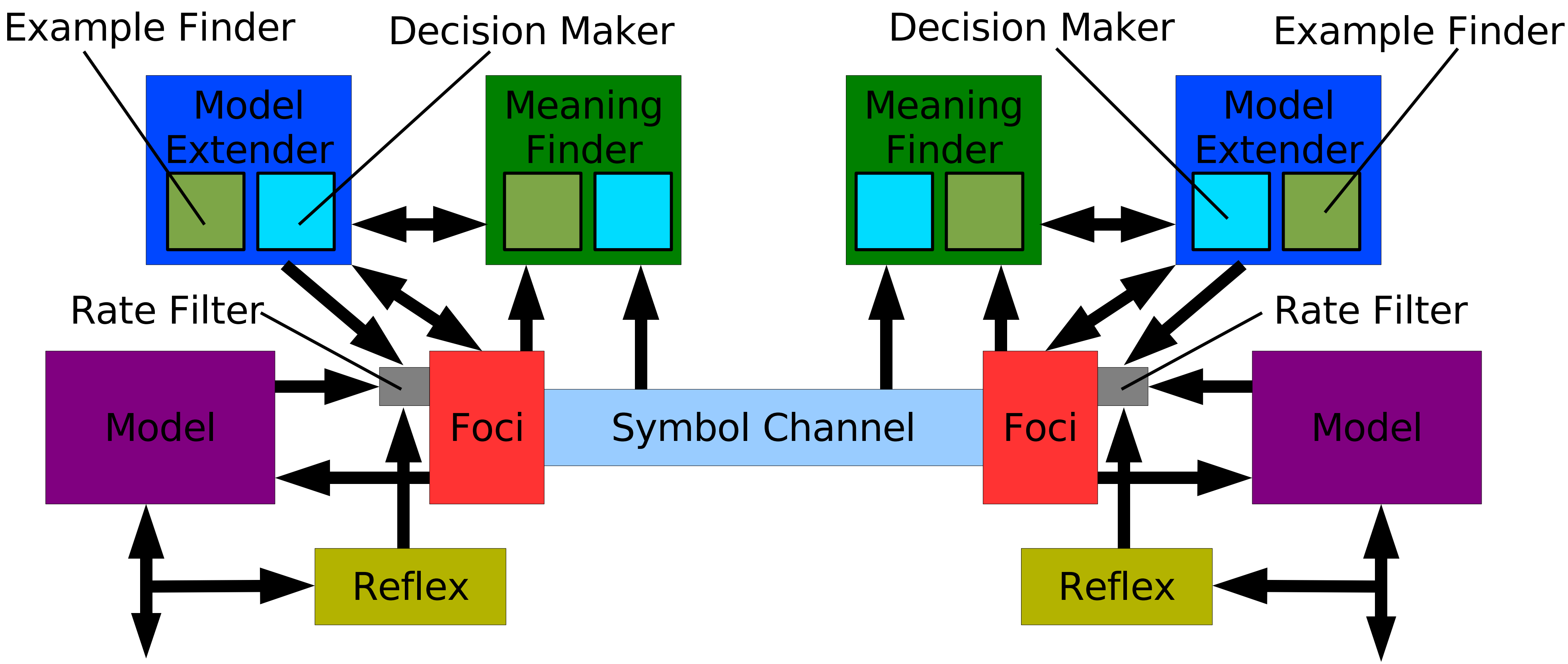
- PN Hierarchy (Beal '03)
  - Sparse node initiation
  - Fast stabilization to even distribution





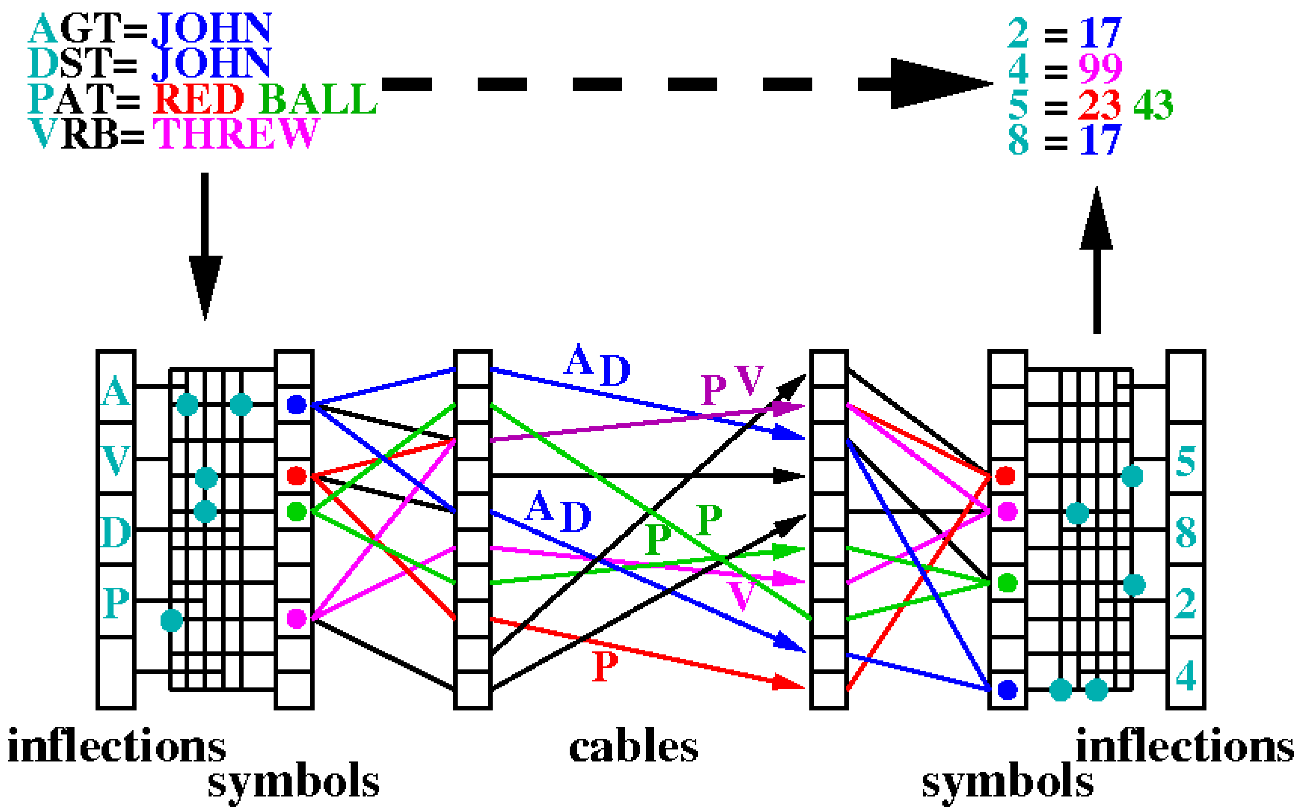
Cognitive Architectures

- How can we build **towards** human-like competence and flexibility?
  - Example: communication bootstrapping



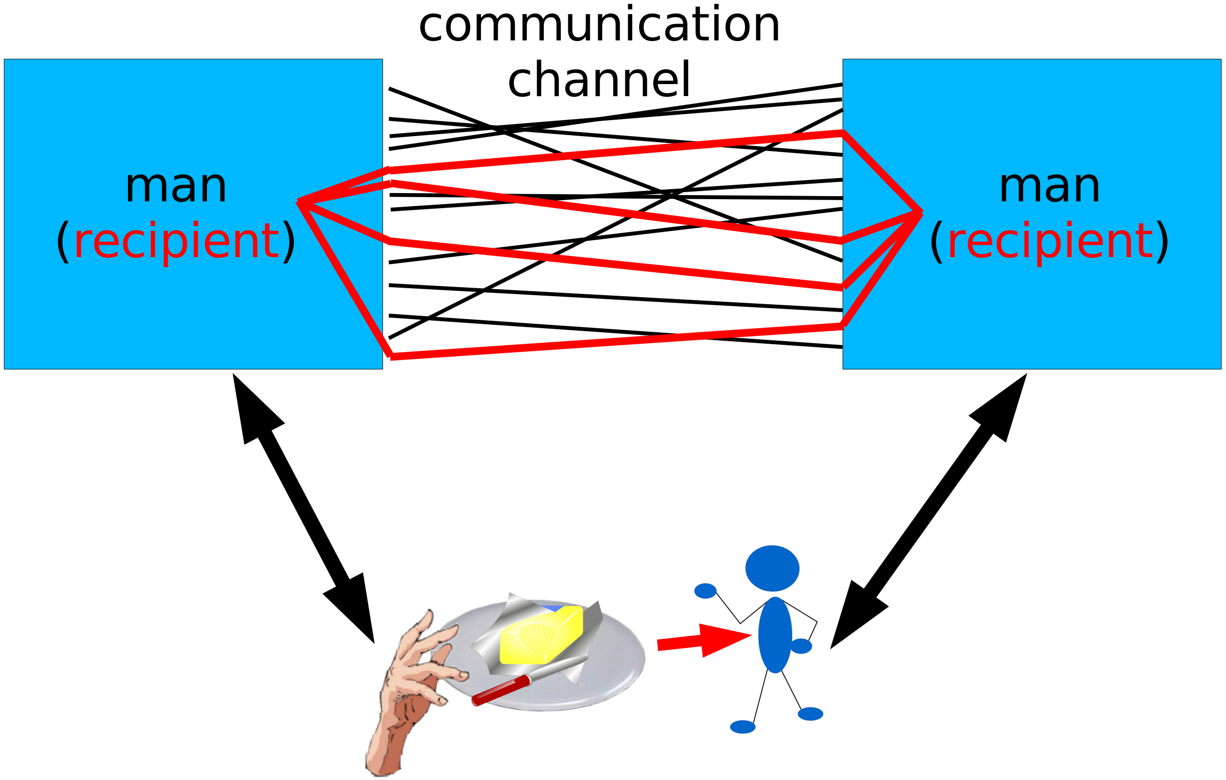
Sparseness: Self-Organization

- Biologically Plausible Symbolic Link:
  - Encodings are sparse pulses on sparse wires
  - Fast organization, burst transmission



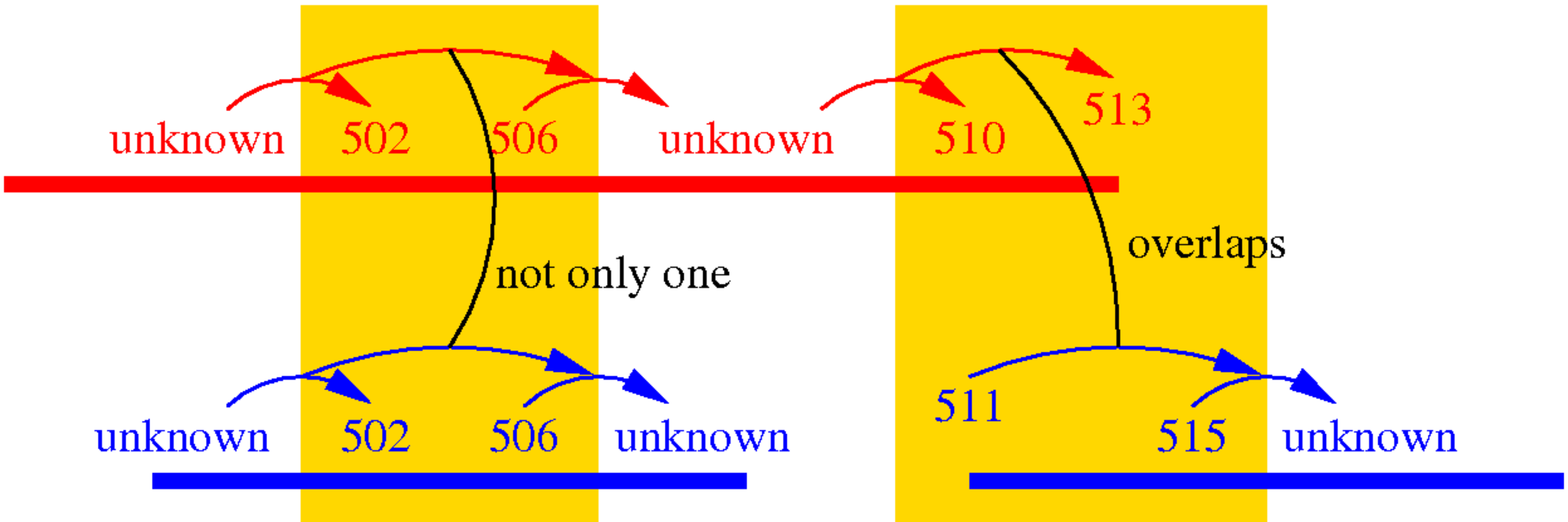
Sparseness: Integration

- Communication Bootstrapping (Beal '02):
  - Sparse sensory input, encodings
  - Fast association, synchronized models



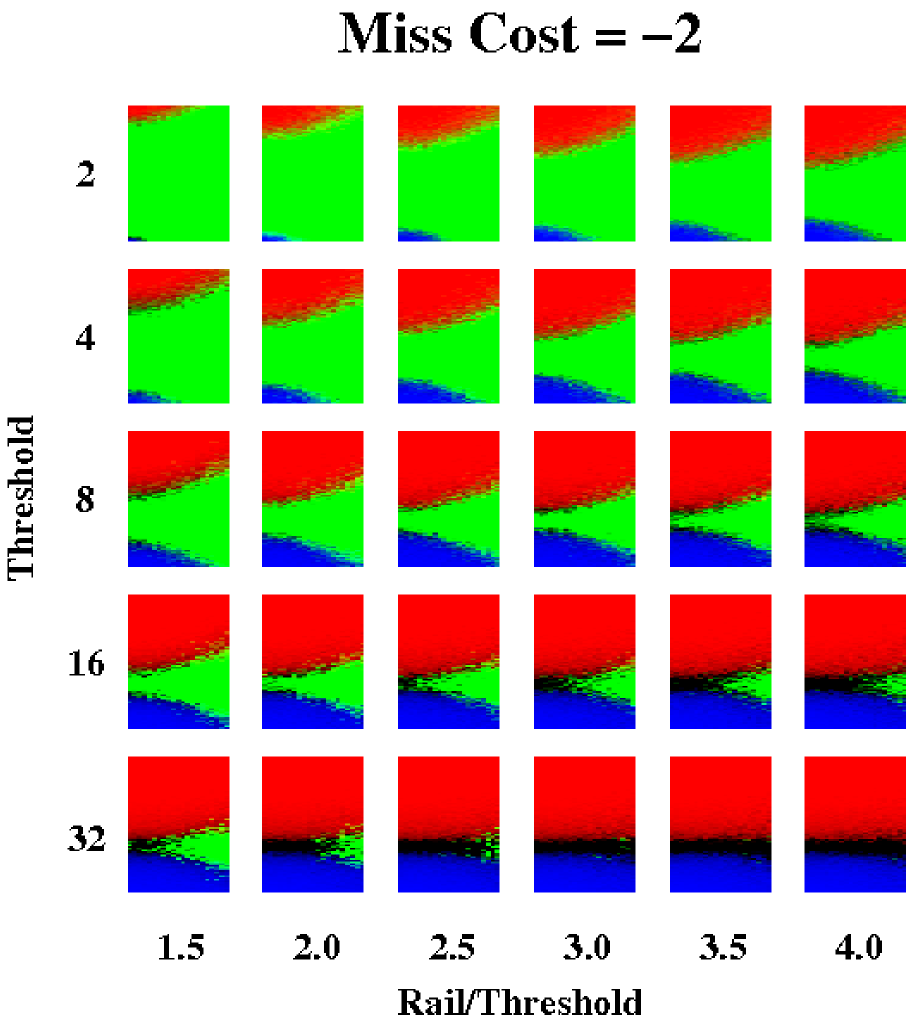
Self-Scaling: Interval Relations

- Incremental Interval Exa. Segmentation:
  - Templates from Allen's time relations
  - Scales by ignoring irrelevancies

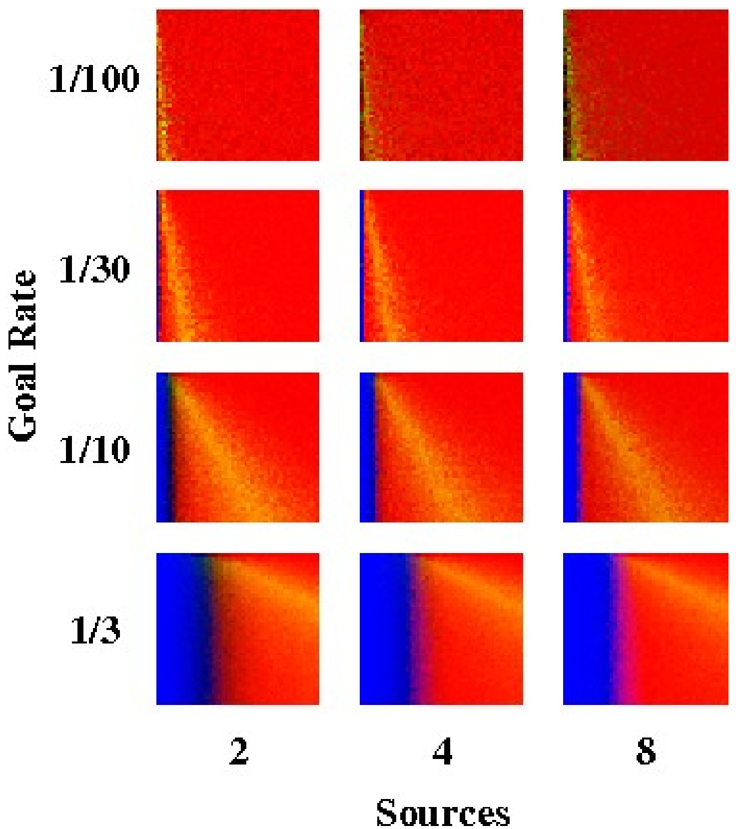


Engineered Insensivity to Parameters and Conditions

Coincidence Detector:

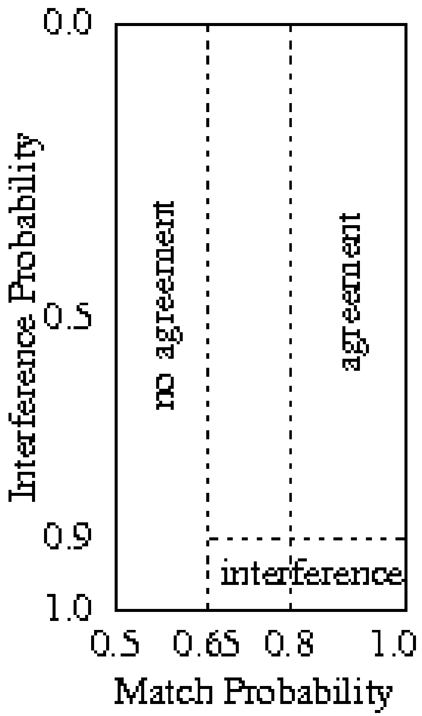


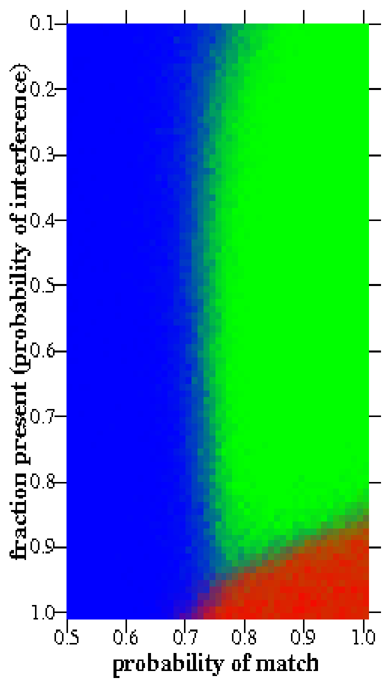
Event Throttle:



Fail. Simp.: Pre-emptive Failure

- Coincidence Detector: if it's not a fast success, it's a failure.
  - Predictable flexibility for signal agreement, BPSL, unidirectional-to-bidirectional link





Putting it all together...

- We now have a toolbox containing:
  - four ways to simplify ugly interactions: self-scaling, sparseness, gradual degradation, failure simplification
  - guidelines for where to use them
  - examples of how to use them in two domains

What's next?

- Refining their application
  - Proto/Amorphous Medium
  - Communication Bootstrapping Architecture
- Where else can we apply them?
- What other tools do we need?