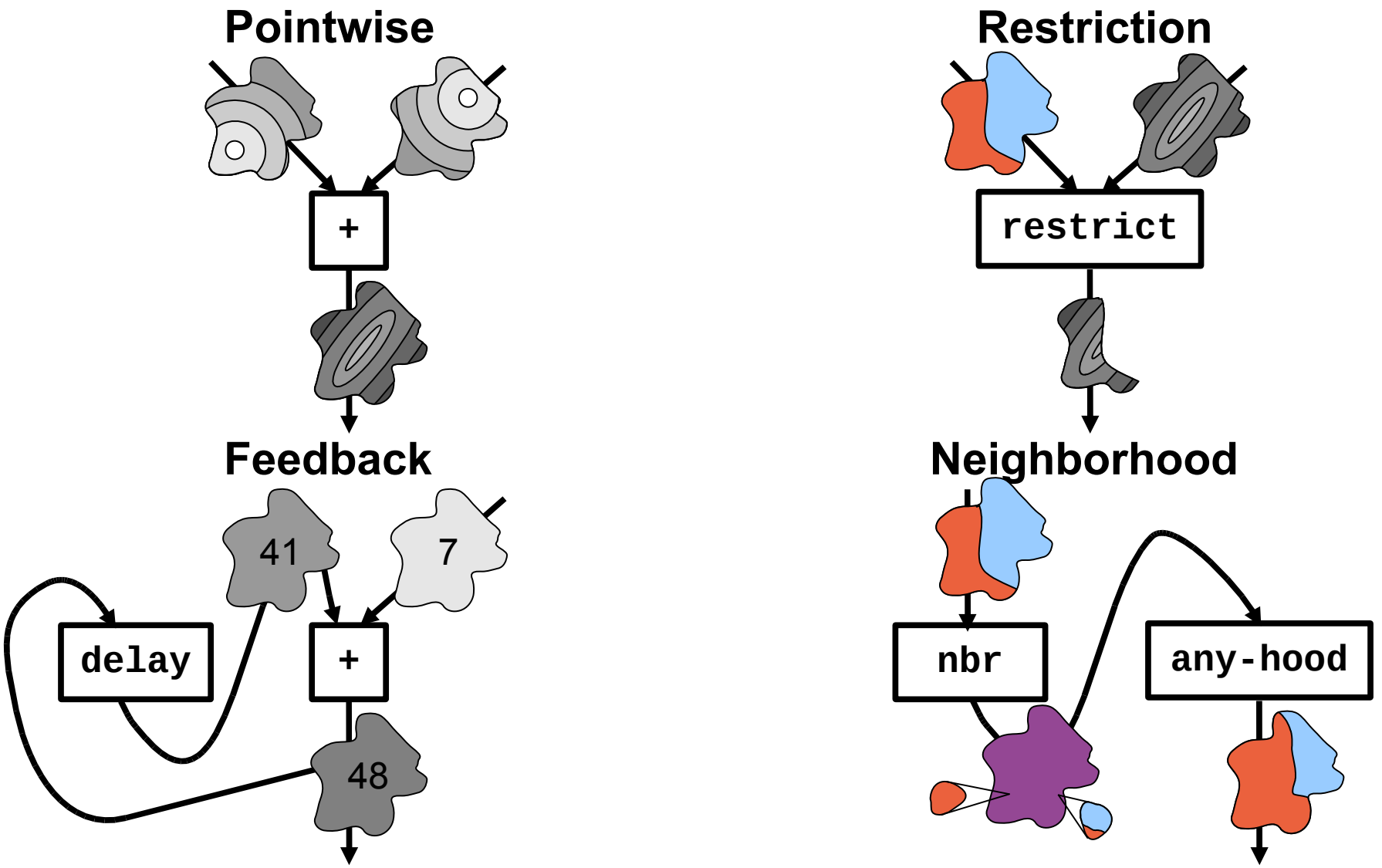
Continuous Space-Time Programs

Jacob Beal

Lecture 2 of 5 on Spatial Computing
ISC-PIF Summer School, 2009

BBN TECHNOLOGIES

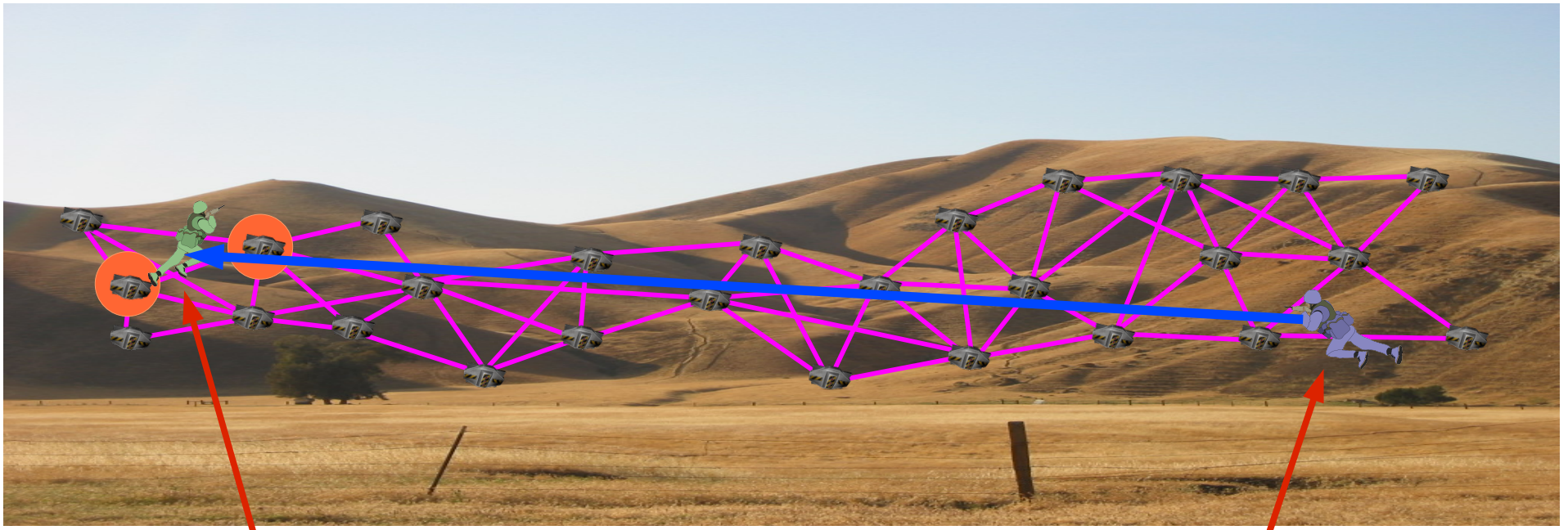
or: global to local in four easy pieces



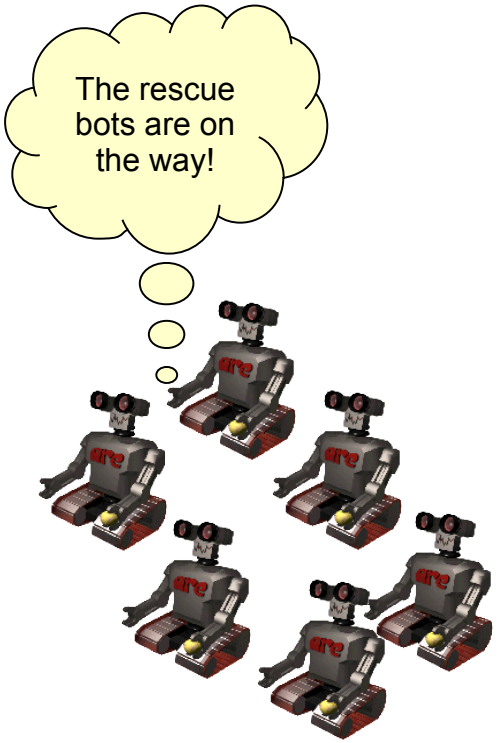
Agenda

- Survey of Spatial Computing Approaches
- Amorphous Medium Abstraction
- Space-Time Primitives
- Compiling from Global to Local

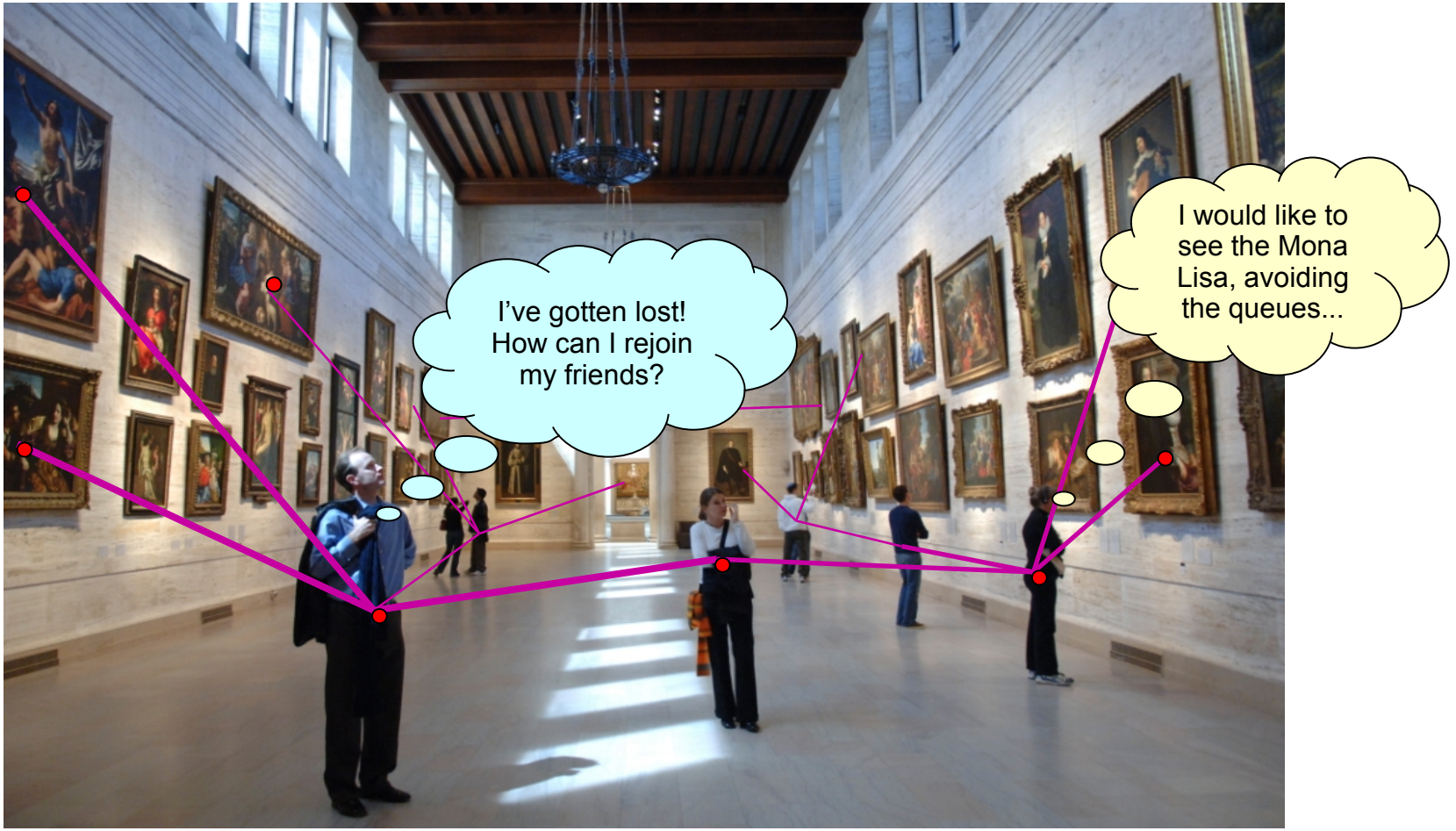
Example: Target Tracking



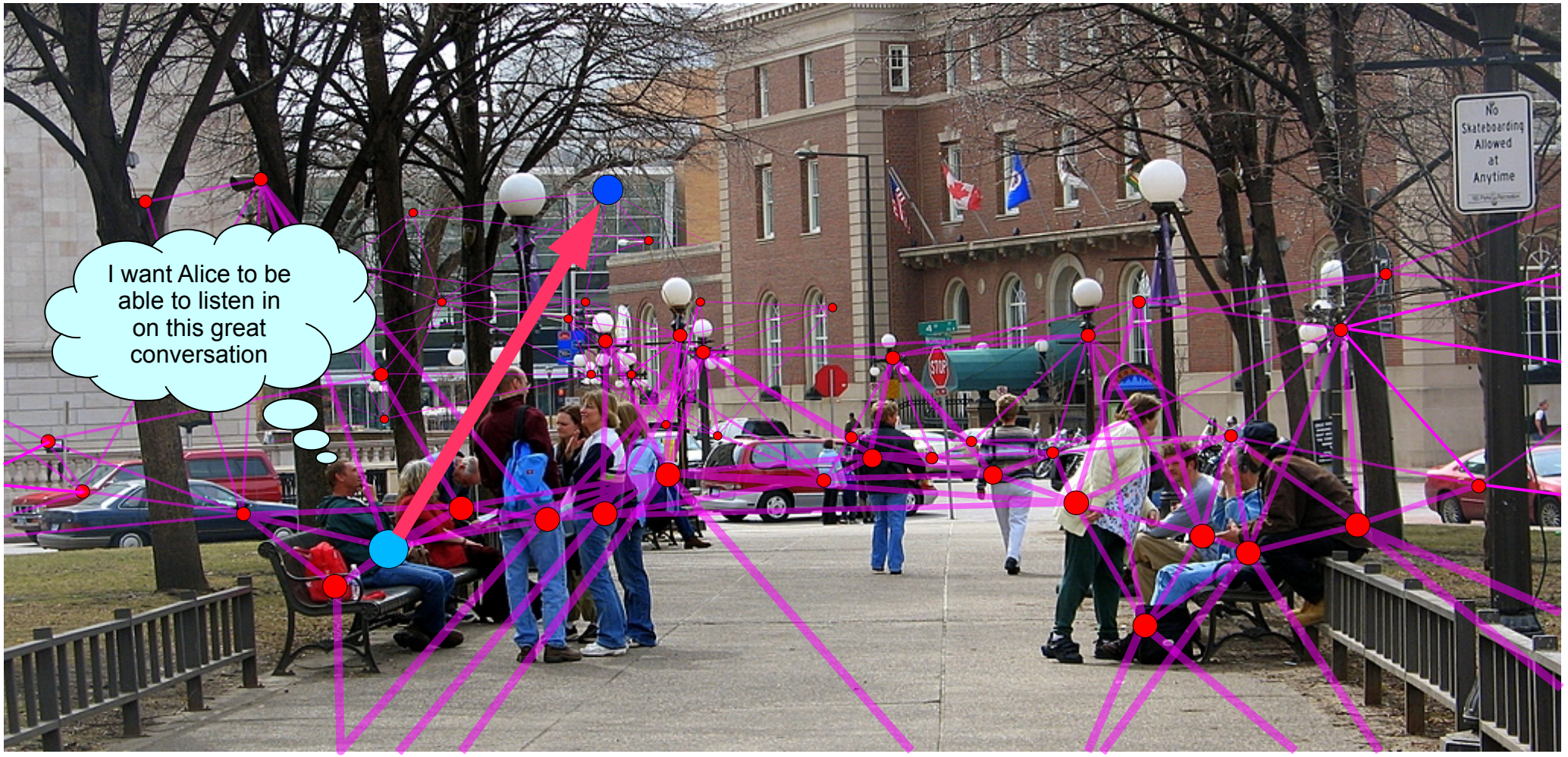
Example: Search & Rescue



Example: Museum Guide



Example: Mobile Streaming



How can we program these?

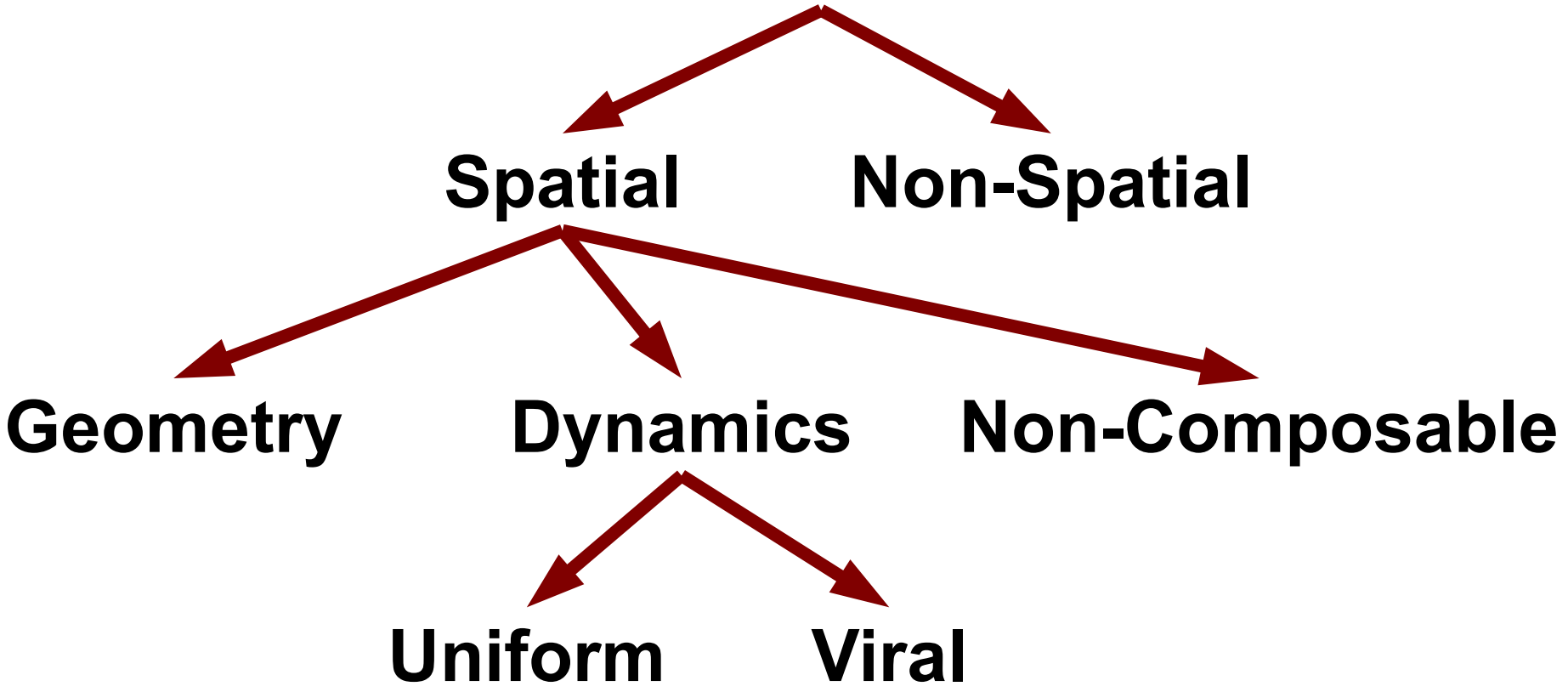
- Desiderata for approaches:
  - Simple, easy to understand code
  - Robust to errors, adapt to changing environment
  - Scalable to potentially vast numbers of devices
  - Take advantage of spatial nature of problems

What is a Language?

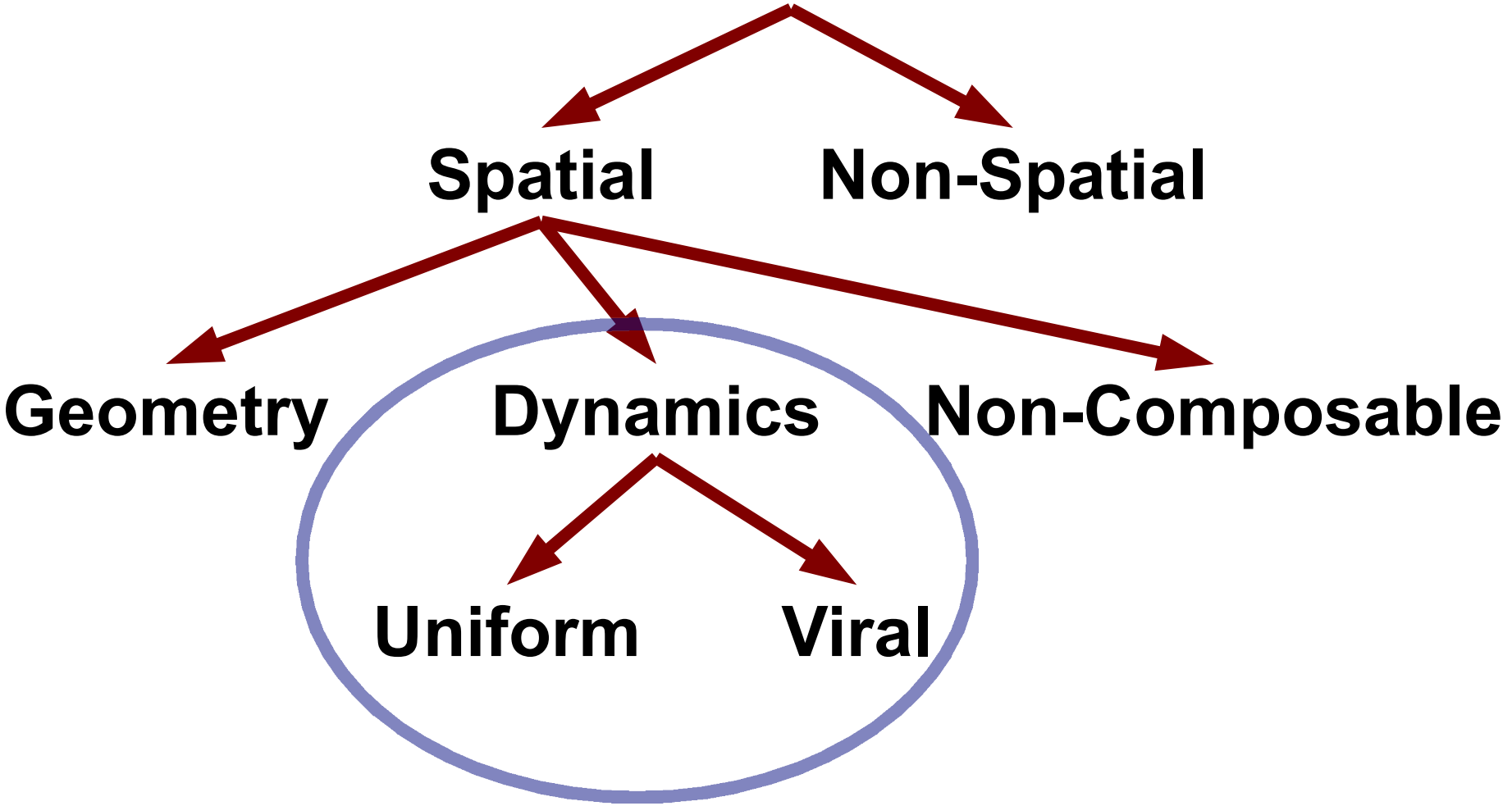
- Standardized library of parts **[Primitives]**
- Rules for building bigger parts by combining smaller parts **[Composition]**
- Mechanism for naming parts and treating them like primitives. **[Abstraction]**

What is explicit and what is implicit?

A Taxonomy of Approaches



A Taxonomy of Approaches



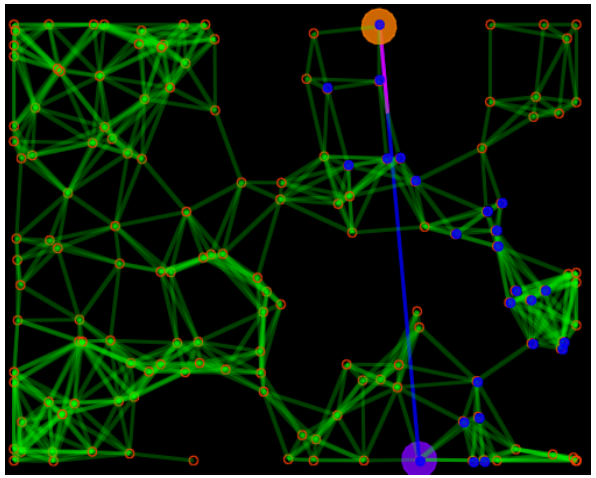
Approaches from Local Dynamics

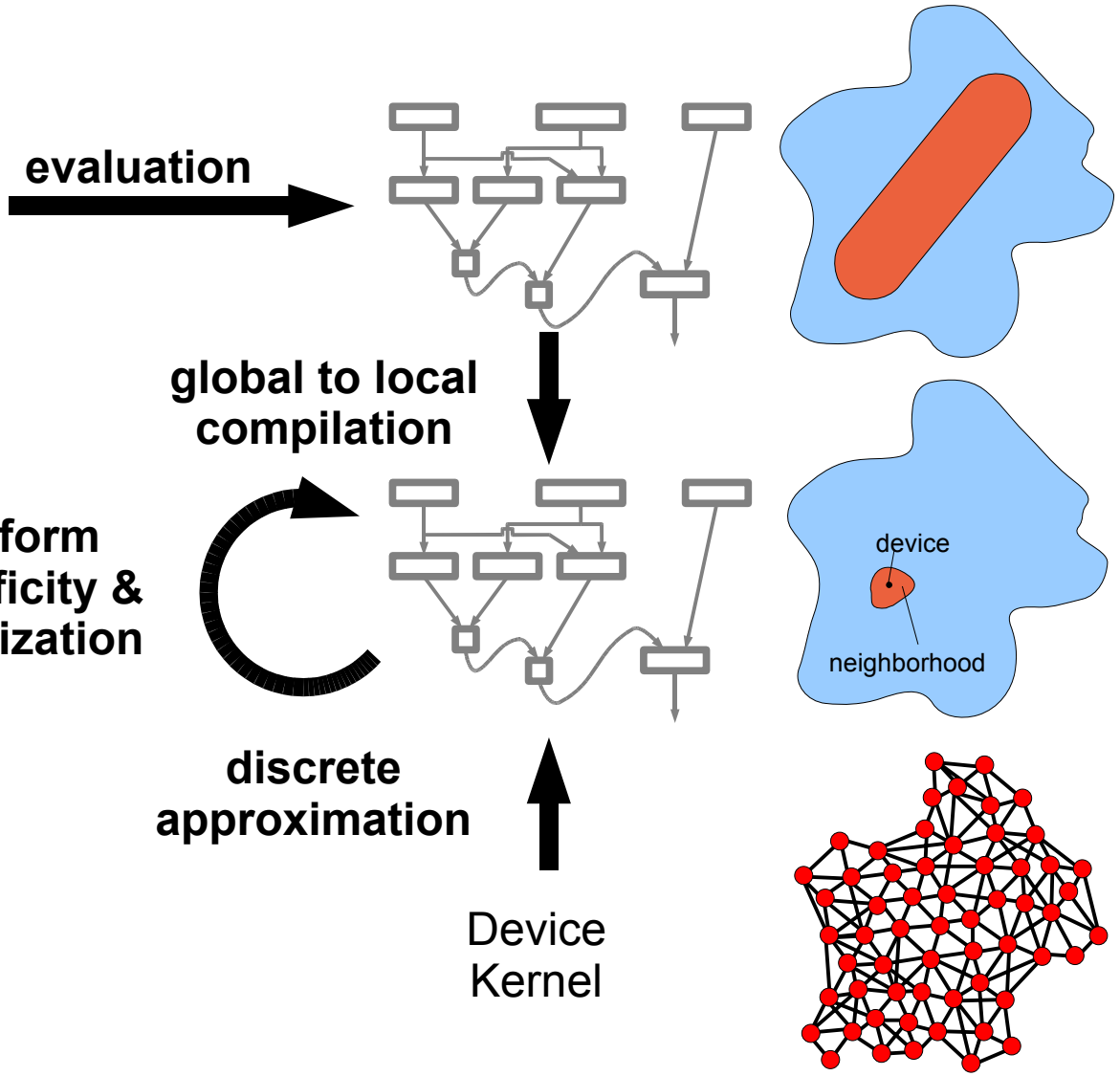
Primitives describe only actions between devices and the neighbors they communicate with.

- Advantages: coherent and correct semantics
- Disadvantages: programmer must figure out how to marshal local dynamics to produce coherent large-area programs

Proto: Computing with Fields

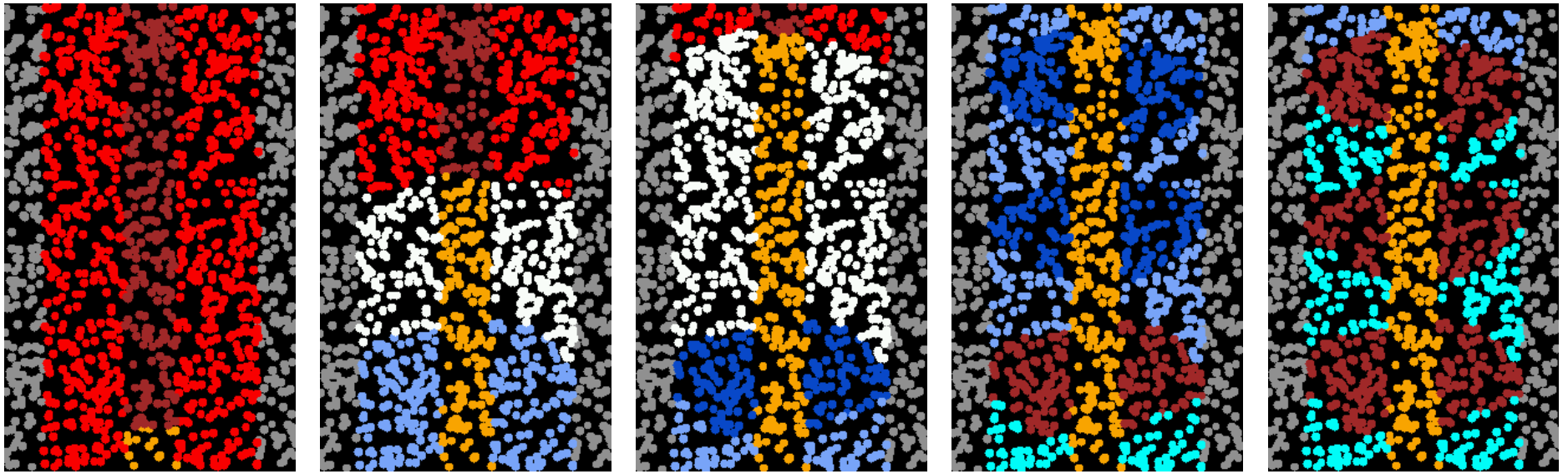
```
(def gradient (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (gradient src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
         (trail (<= (+ (gradient src)
                       (gradient dst))
                    d)))
    (dilate trail width)))
```





[Beal & Bachrach '06]

Microbial Colony Language



- Marker diffusion & decay, events
- Closely targetted at engineered bacteria

[Weiss et al., '98]

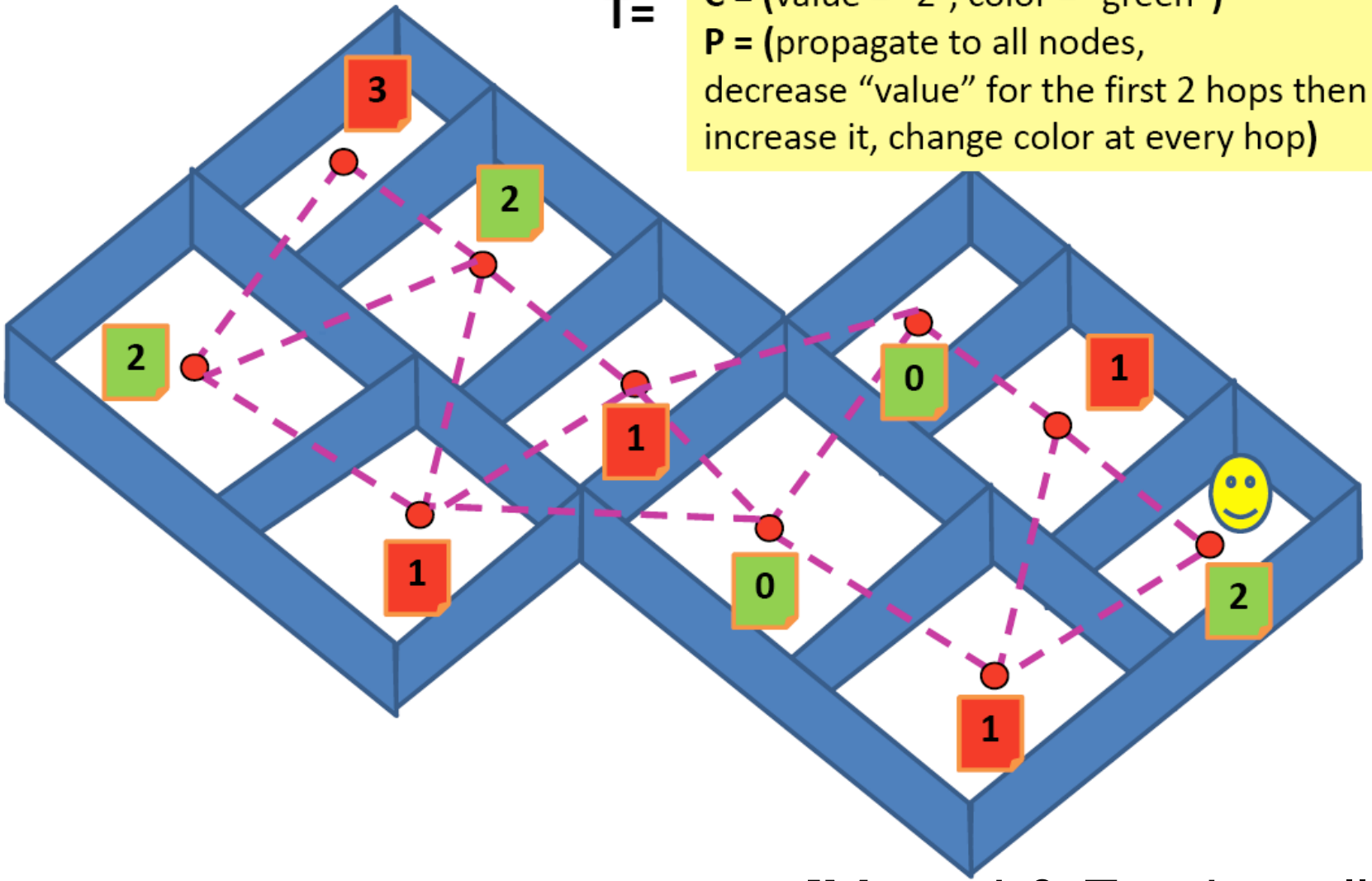
Other Uniform Approaches

- LDP [De Rosa, et al., '08], Meld [Ashley-Rollman '07]
  - Distributed logic programs
  - Local resolution leads to long-distance properties

TOTA: Viral tuples

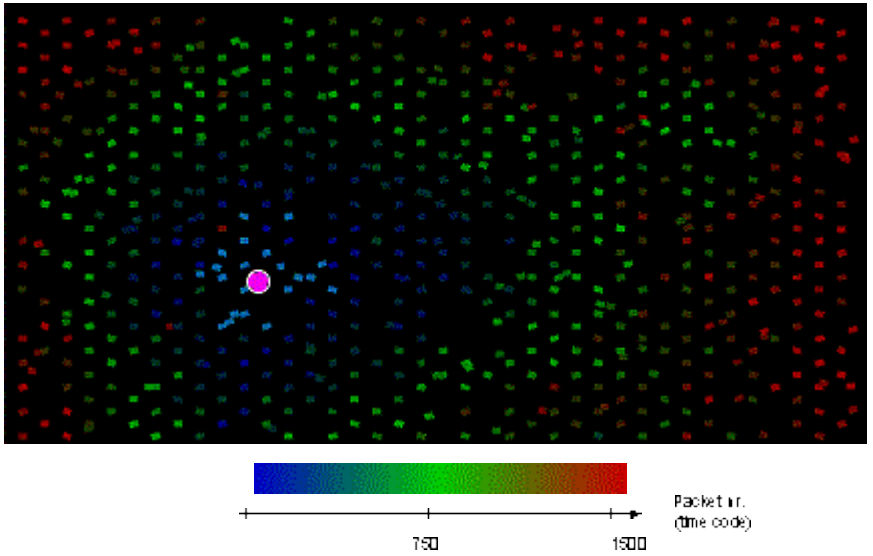
T=

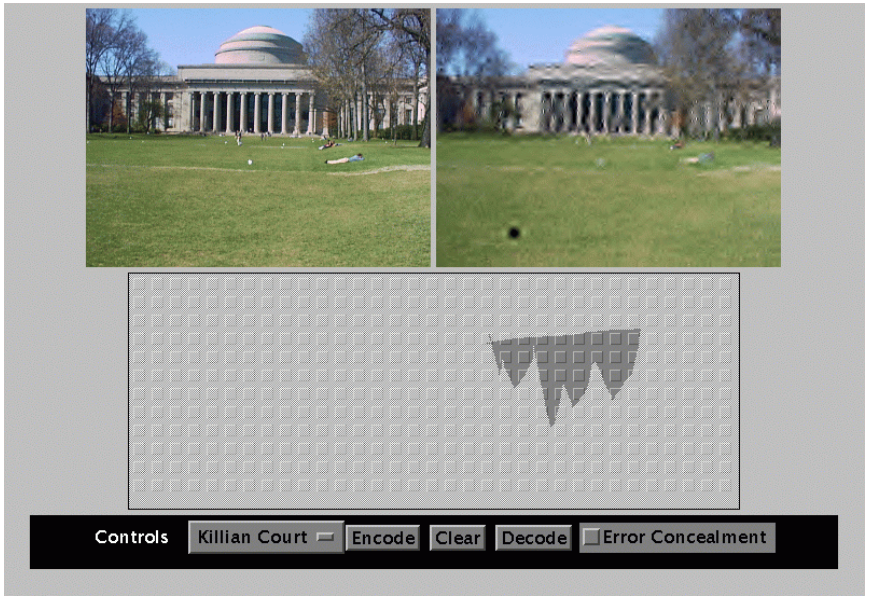
C = (value = “2”, color = “green”**)**
P = (propagate to all nodes,
decrease “value” for the first 2 hops then
increase it, change color at every hop**)**



[Mamei & Zambonelli, '08]

Paintable Computing





- Consistent transfer, view of neighbor data
- Code for install, de-install, transfer-granted, transfer-denied, update

[Butera, '02]

Other Viral Approaches

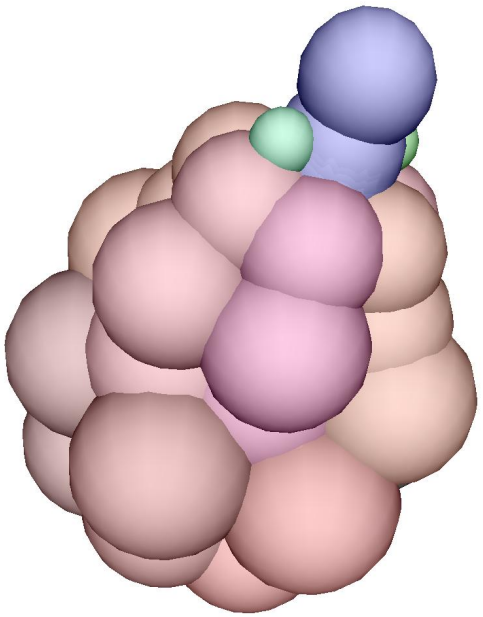
- Smart Messages [Kang et al, '04]
  - Execution migrates to nodes of interest, found via self-routing code packets
- RGLL [Sutherland, '03]
  - Code for arrival, tick, collision, departure
  - Communication via collision

Approaches from Geometry

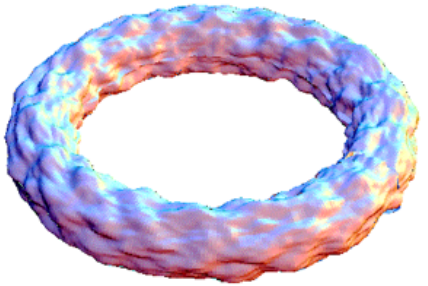
Primitives describe large-scale geometric regions (e.g. “all devices on the left hill”)

- Advantages: coherent, easy to specify large-scale programs
- Disadvantages: generally easy to accidentally specify programs that cannot be executed correctly

MGS



Meristem formation



Turing pattern on torus

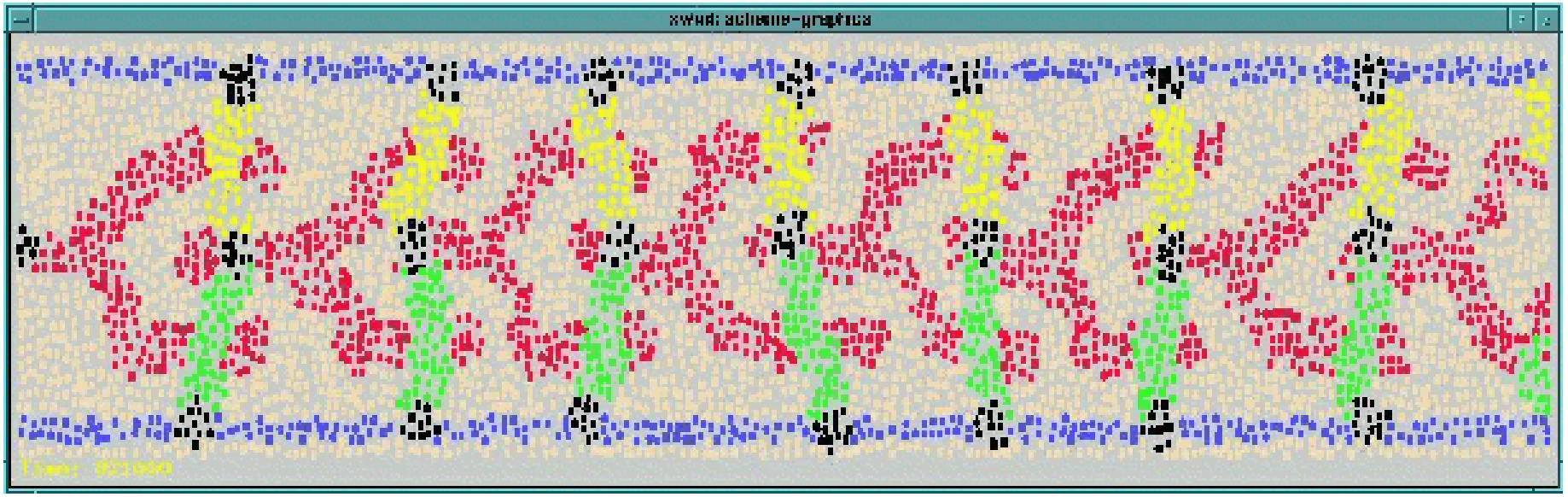
[Michel, Giavitto, Spicher, 2004]

Regiment

- Streaming collection of data from regions
  - Spatial primitives:
    - K-hop neighborhood
    - K-nearest nodes
  - Composition:
    - Union/Intersection
    - Map/Filter
- Distributed execution as a compiler optimization

[Newton & Welsh, '04]

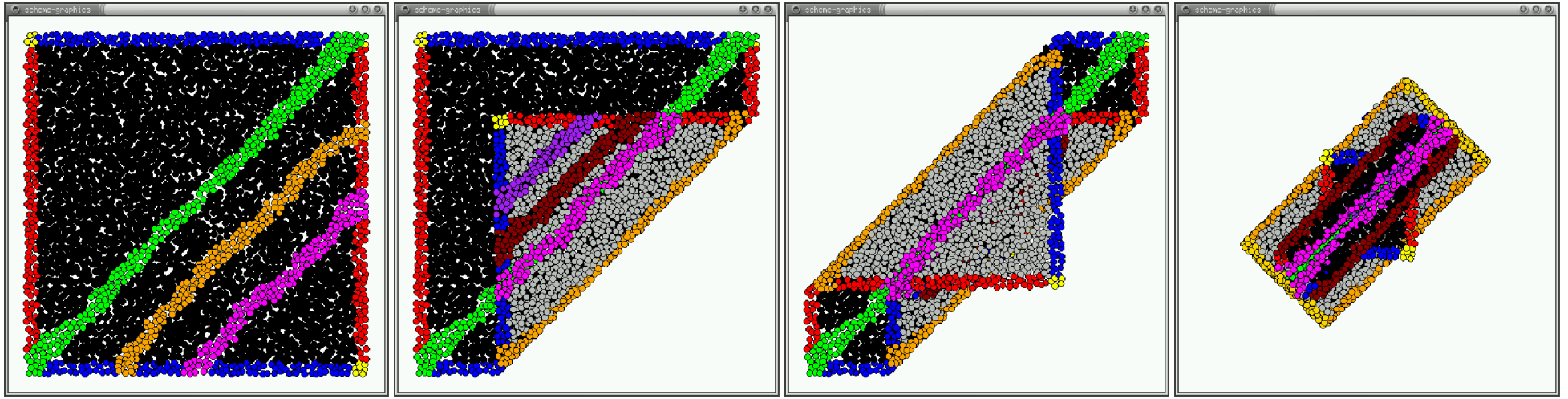
Growing Point Language



- Botanical growing points, chemical tropism
- Can construct arbitrary planar graphs

[Coore, '99]

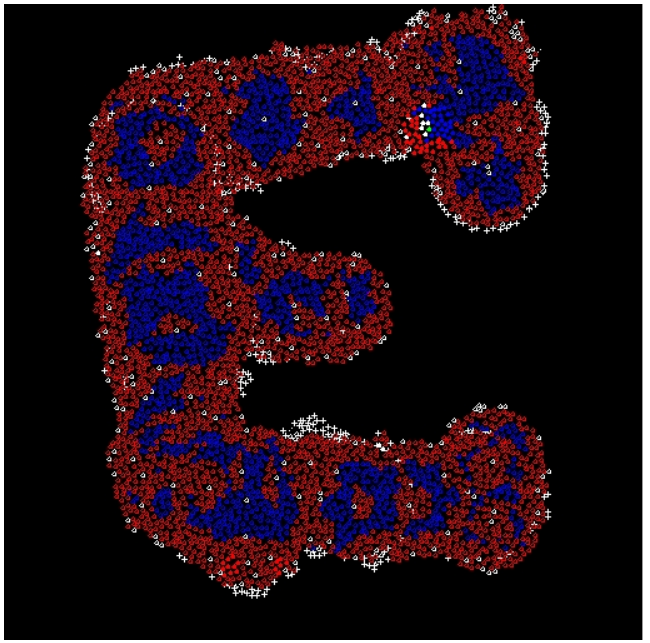
Origami Shape Language

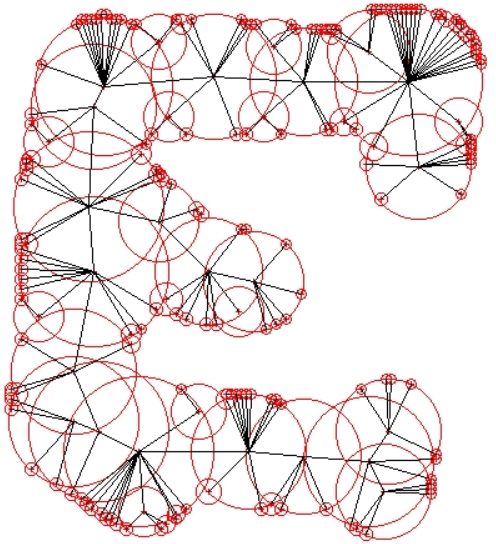


- Geometry and folding sequence
  - Huzita's 6 axioms (e.g. fold Line-1 onto Line-2)
- Predicts *drosophila* morphological variation

[Nagpal, '01]

Growing 2D Shapes





- Grows from single point, fills space with cells
- Scaffolding garbage collects via apoptosis

[Kondacs, '03]

Other Geometric Approaches

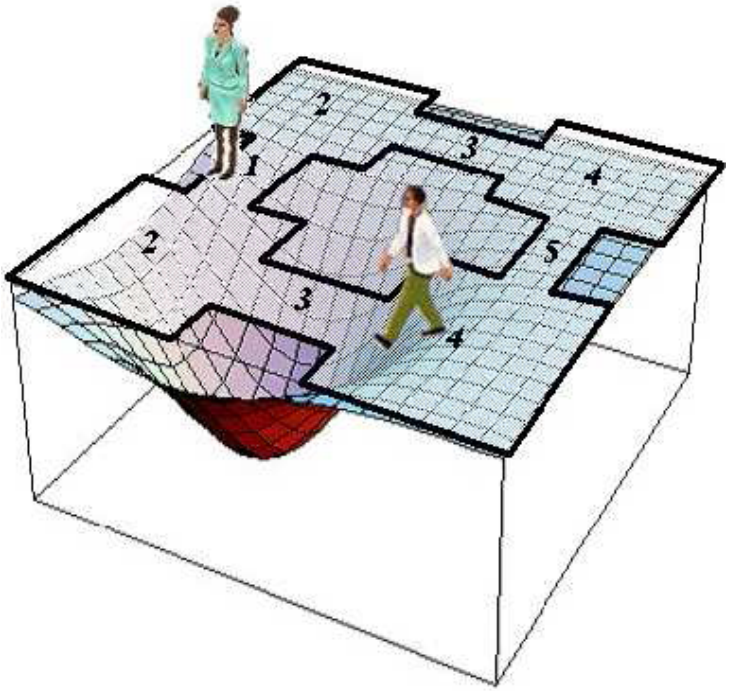
- Spatial Programming [Borcea et al., '04]
- EgoSpaces [Julien & Roman, '06]
- SpatialViews [Ni et al., '03]
- Spidey [Luca & Gian, '06]
- Abstract Regions [Welsh & Mainland, '04]

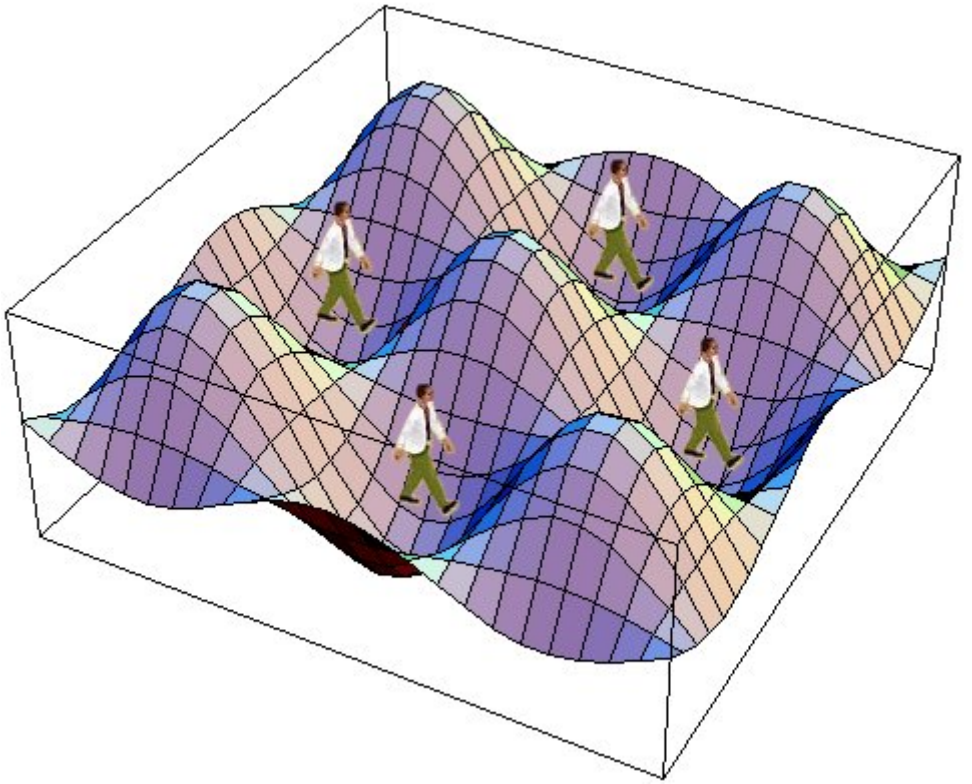
Non-Composable Approaches

Algorithms and techniques, generally based on geometry, but not part of a system of composable parts

- Advantages: powerful spatial ideas for that are good for inclusion in code libraries
- Disadvantages: developed as stand-alone ideas, and may have limited composability

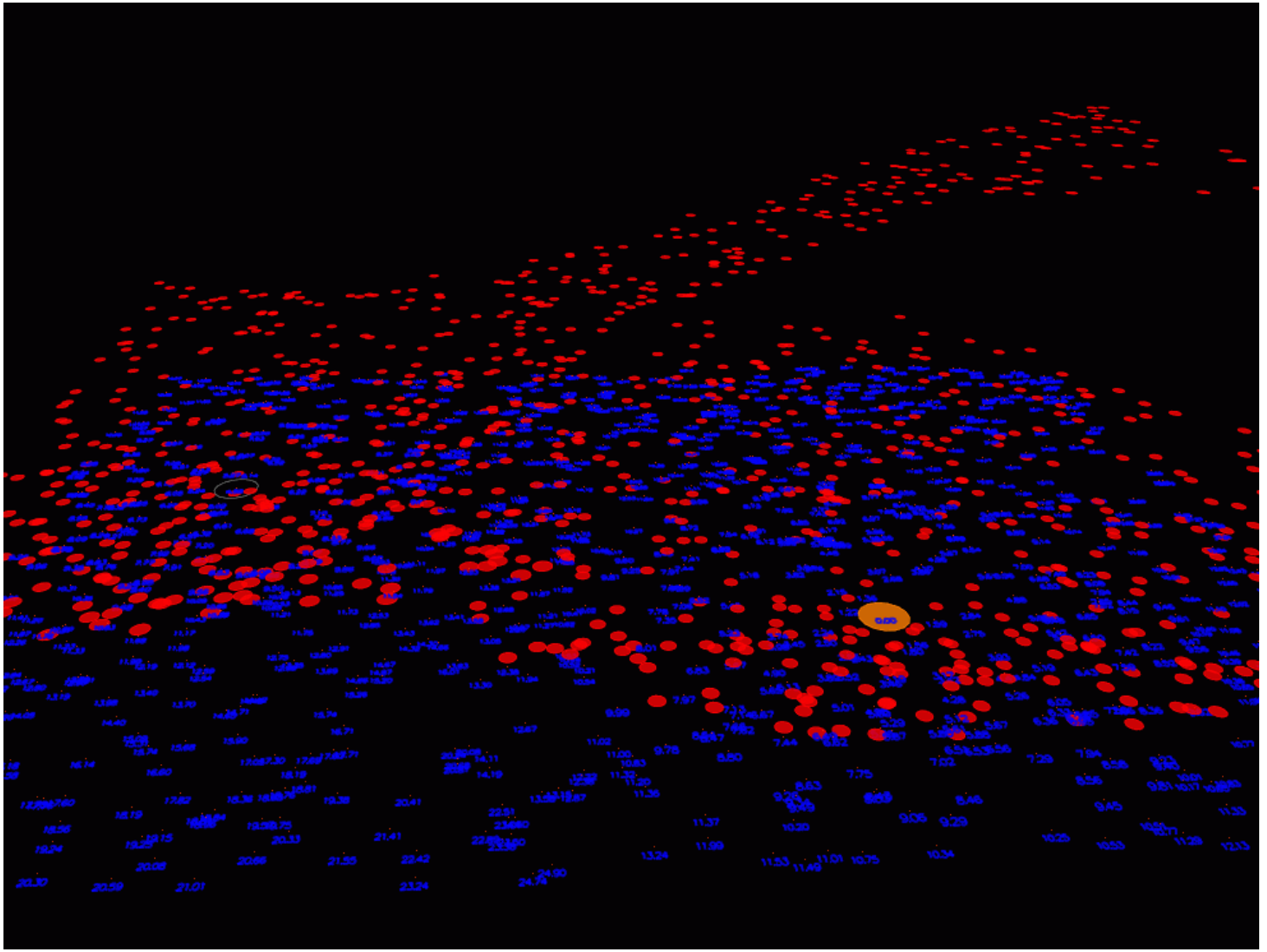
Field-Based Coordination





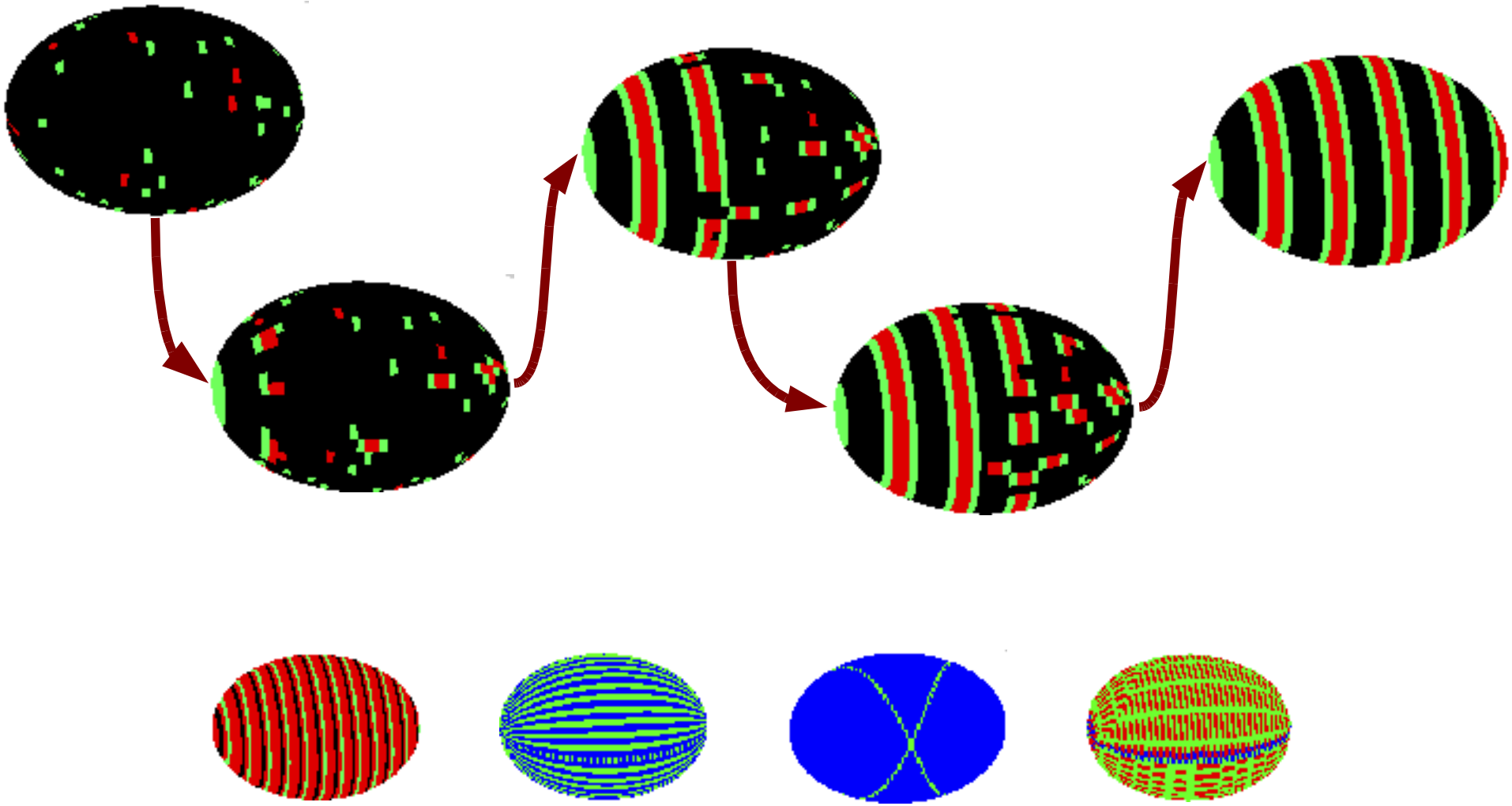
[Mamei & Zambonelli, '06]

Self-Healing Gradients



[Clement & Nagpal '03; Butera '02; Beal et al., '08]

Local Check-Schemes



[Yamins, '08]

Other Non-Composable Approaches

- hood [Whitehouse, et. al., '04]
  - nesC library for interacting with neighbors
- “Stupid Robot Tricks” [McLurkin '04]
  - Swarm behaviors intended mainly for time-wise multiplexing.
- *Countless one-shot systems...*

Significant Non-Spatial Approaches

- “roll-your-own” (e.g. C/C++)
- TinyDB [Madden et al., '05]
  - Distributed database queries for sensor networks
- Kairos [Gummadi et al., '05]
  - Distributed graph algorithms
- WaveScript [Newton et al., '08]
  - Distributed streaming language
  - Follow-on to Regiment w/o the spatial primitives

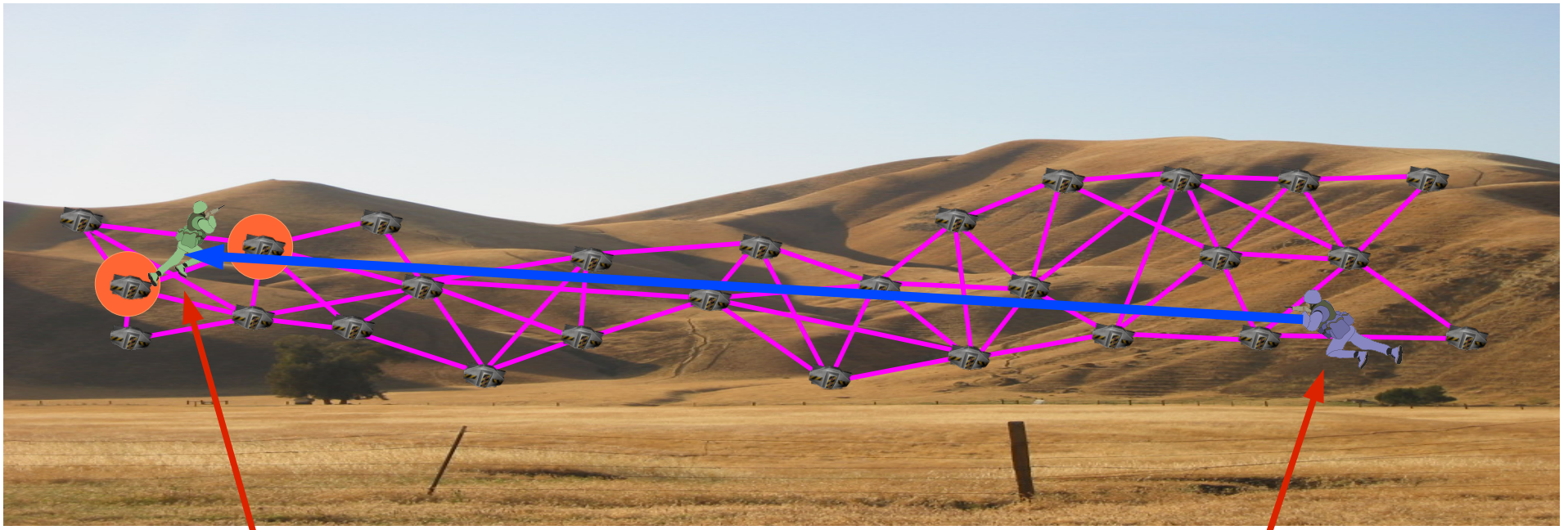
Summary

- Many approaches exist to programming pervasive applications for spatial computers
- Only approaches based on local dynamics currently offer predictable composition, correct execution, and spatial primitives
- Challenge: obtaining long-range coherent behavior from local dynamics

Agenda

- Survey of Spatial Computing Approaches
- **Amorphous Medium Abstraction**
- Space-Time Primitives
- Compiling from Global to Local

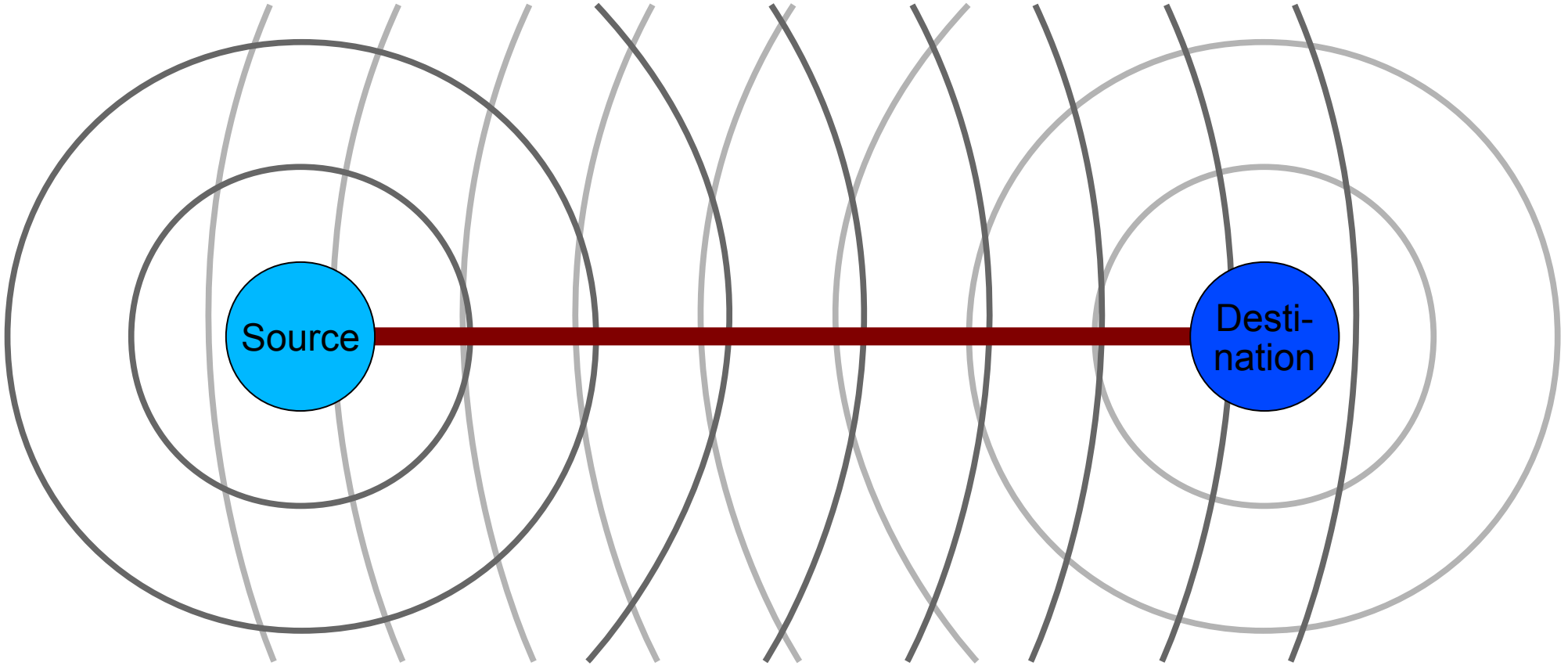
Example: Target Tracking



Geometric Program: Channel

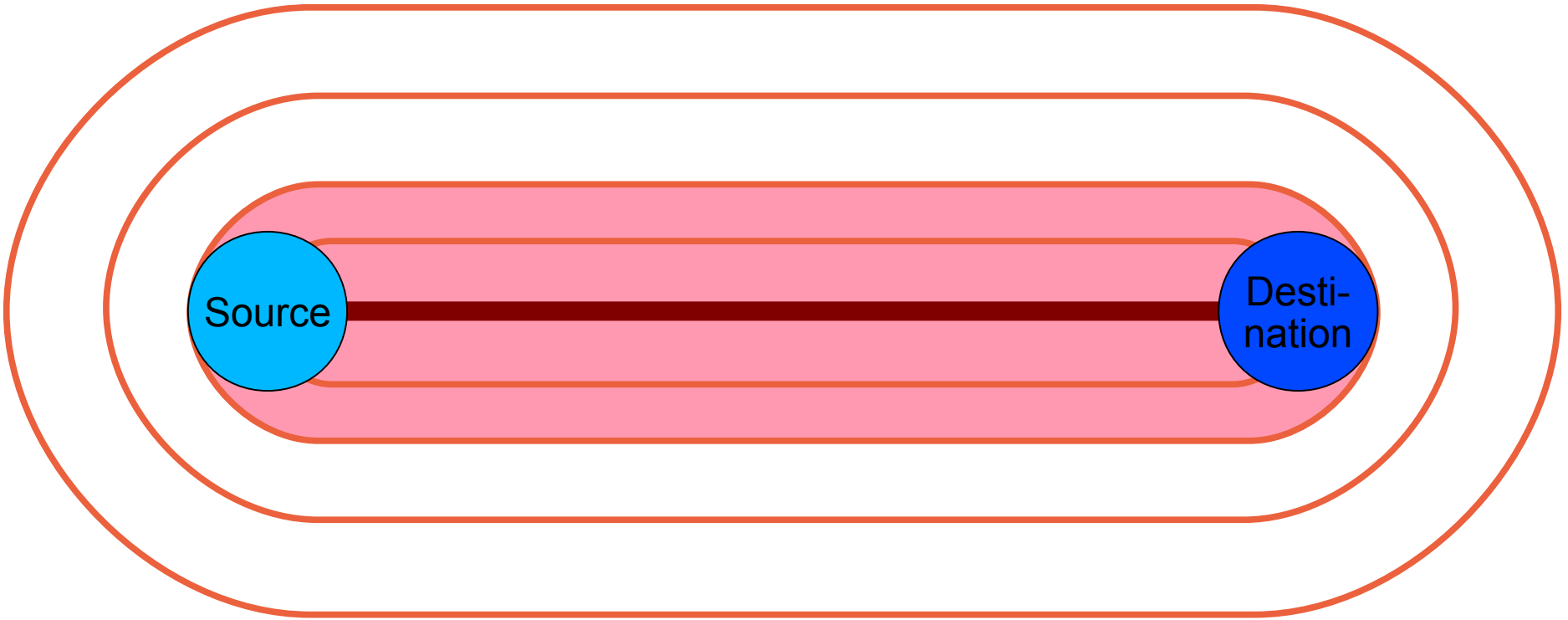
(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel



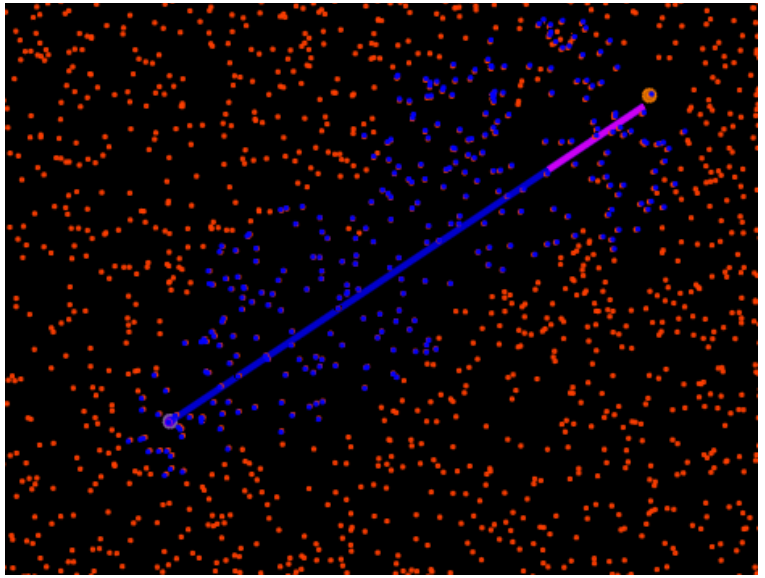
(cf. Butera)

Geometric Program: Channel

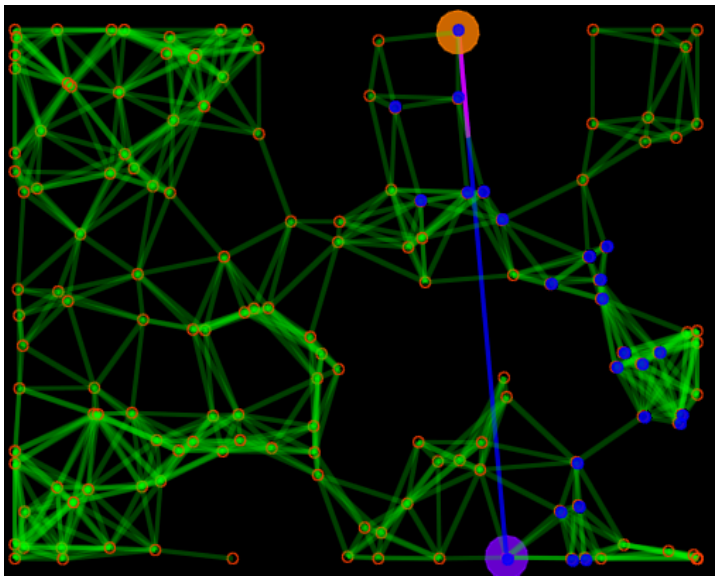
(cf. Butera)

Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

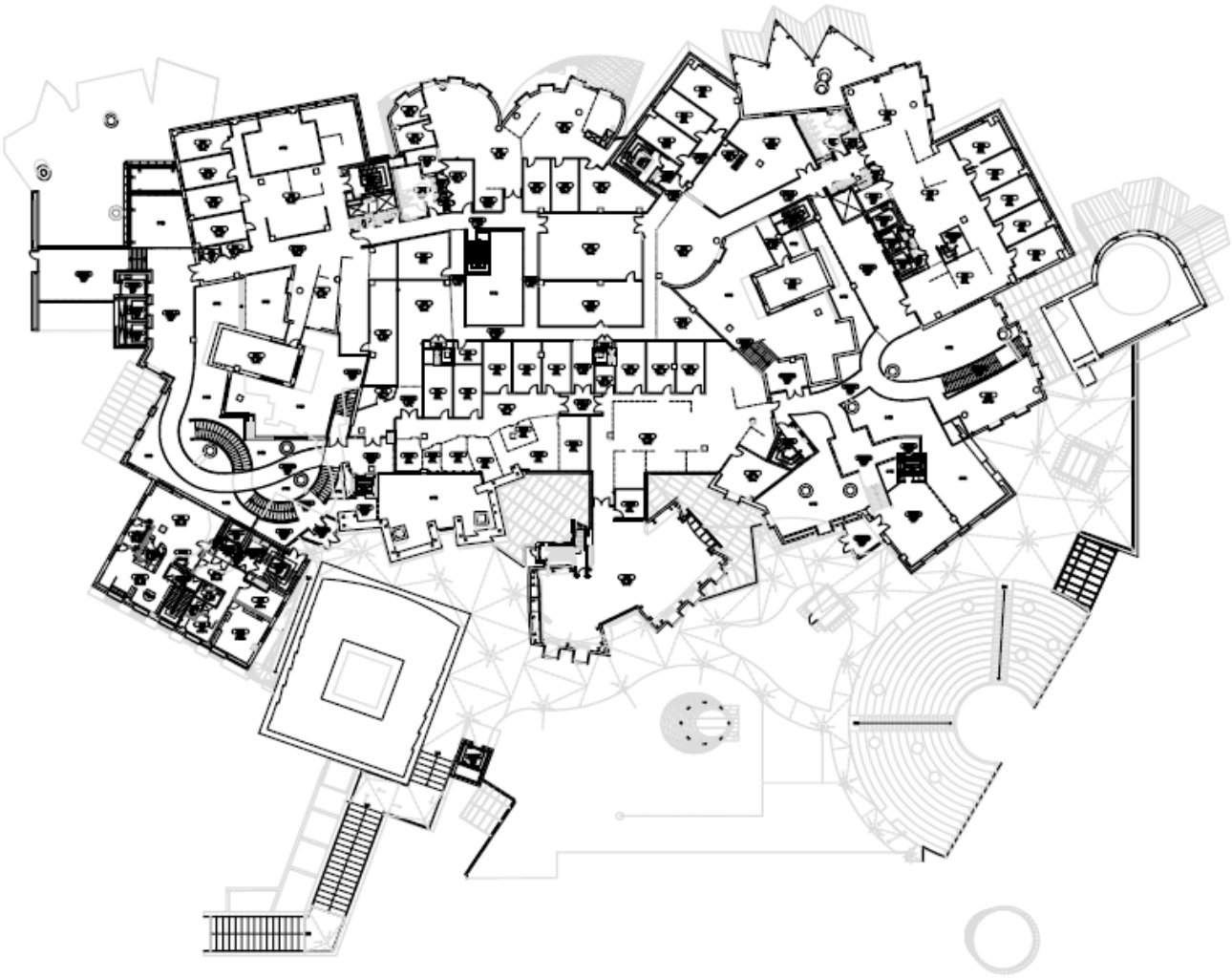


2000 devices

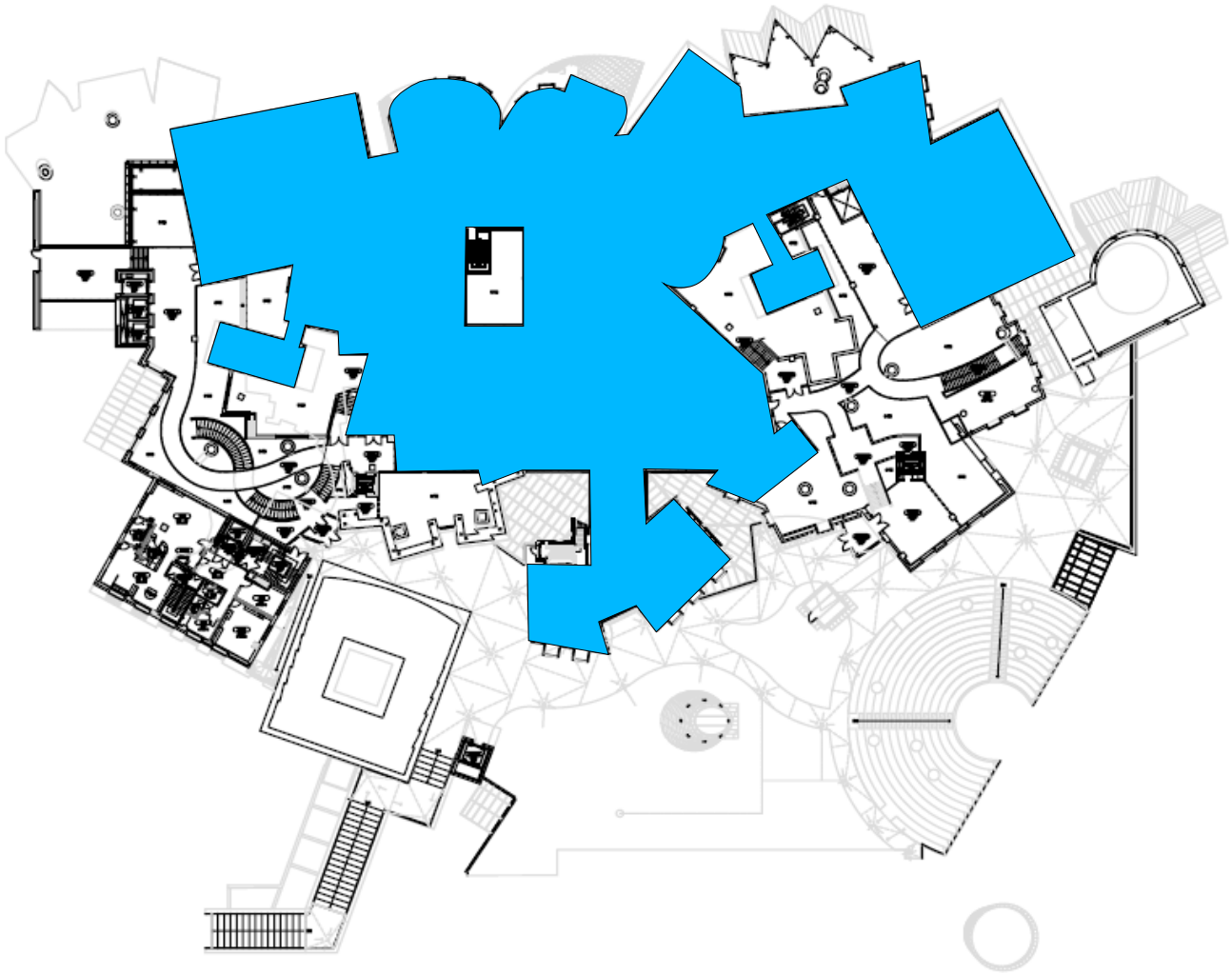


150 devices

Amorphous Medium Approach



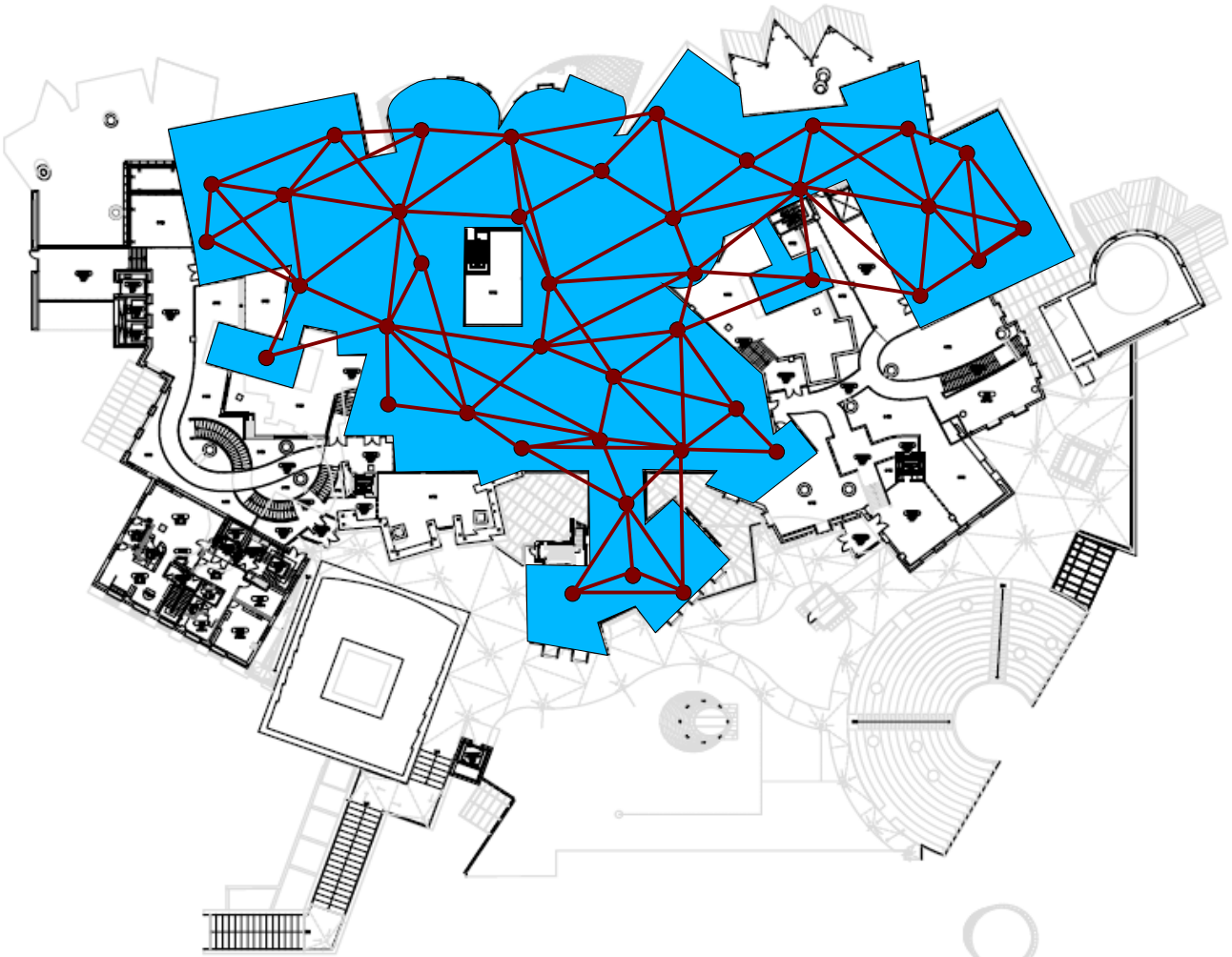
Amorphous Medium Approach



Amorphous Medium Approach

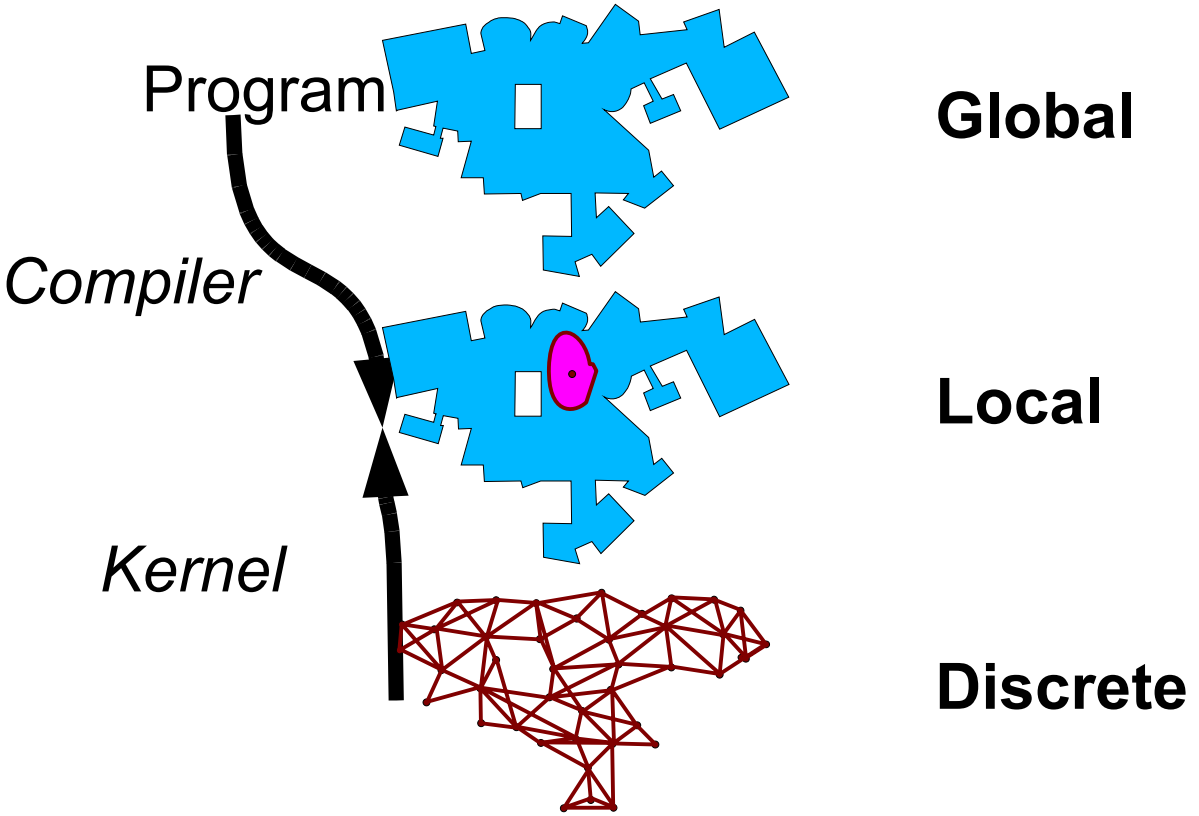
Program the space... approximate with a network

Amorphous Medium Approach

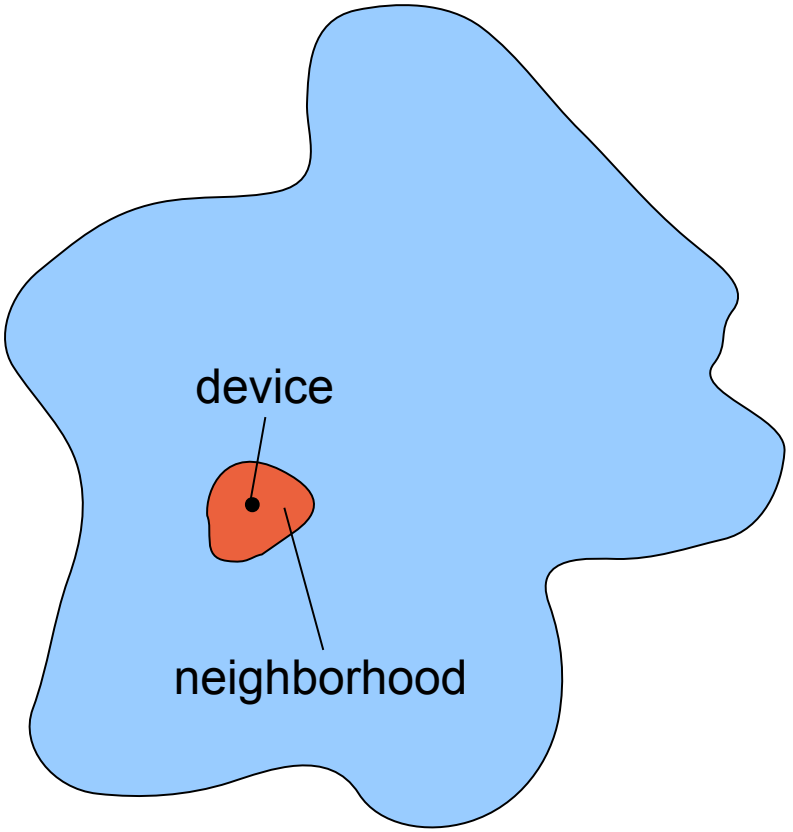


The discretization hardly matters!

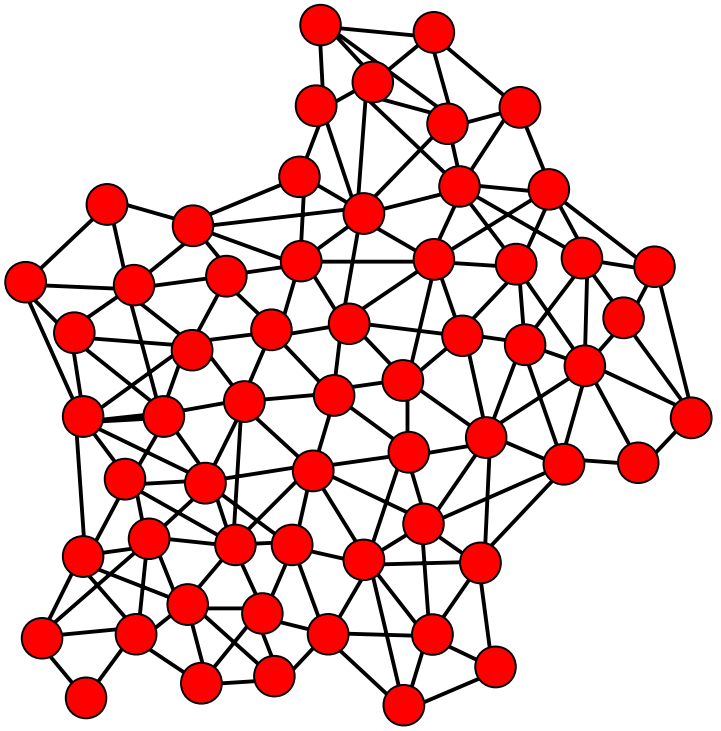
Global v. Local v. Discrete



Amorphous Medium



- Continuous space & time
- Infinite number of devices
- See neighbors' past state



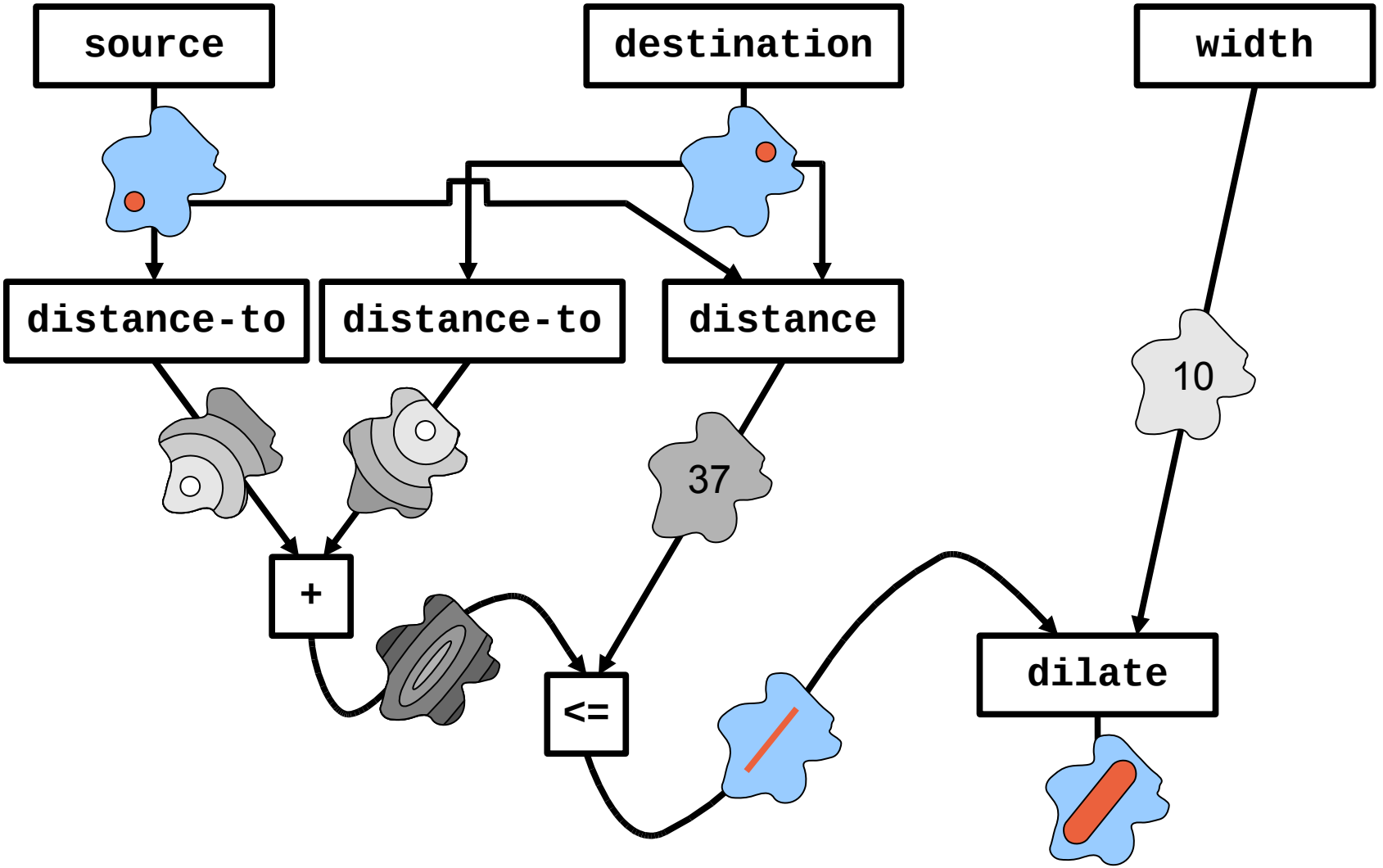
Approximate with:

- Discrete network of devices
- Signals transmit state

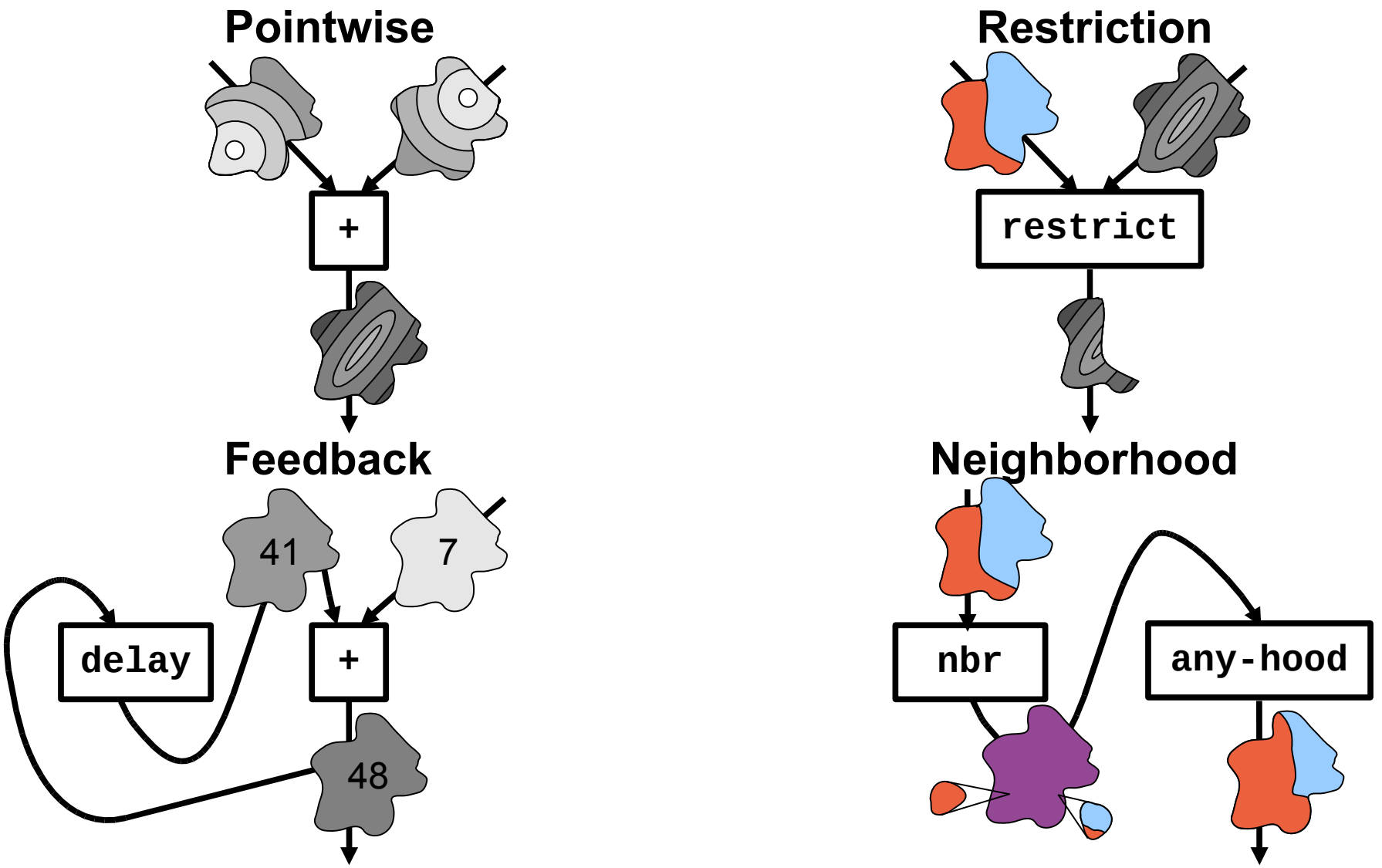
Agenda

- Survey of Spatial Computing Approaches
- Amorphous Medium Abstraction
- **Space-Time Primitives**
- Compiling from Global to Local

Computing with fields

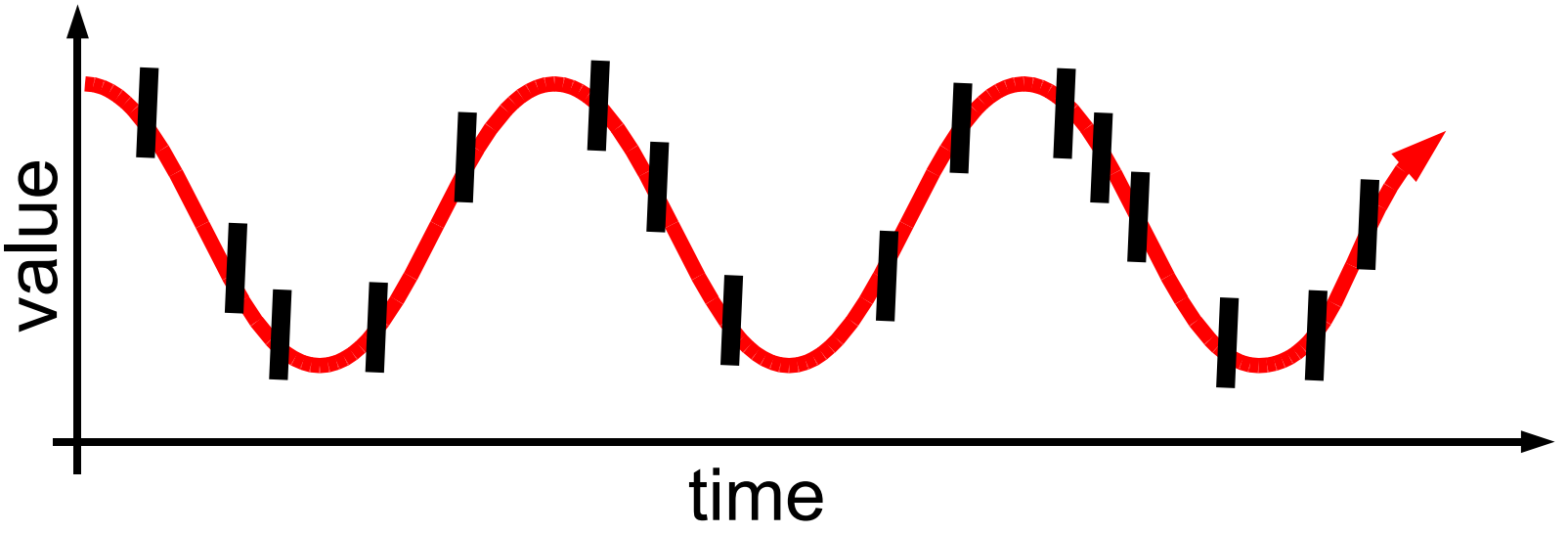


Four Families of Primitives

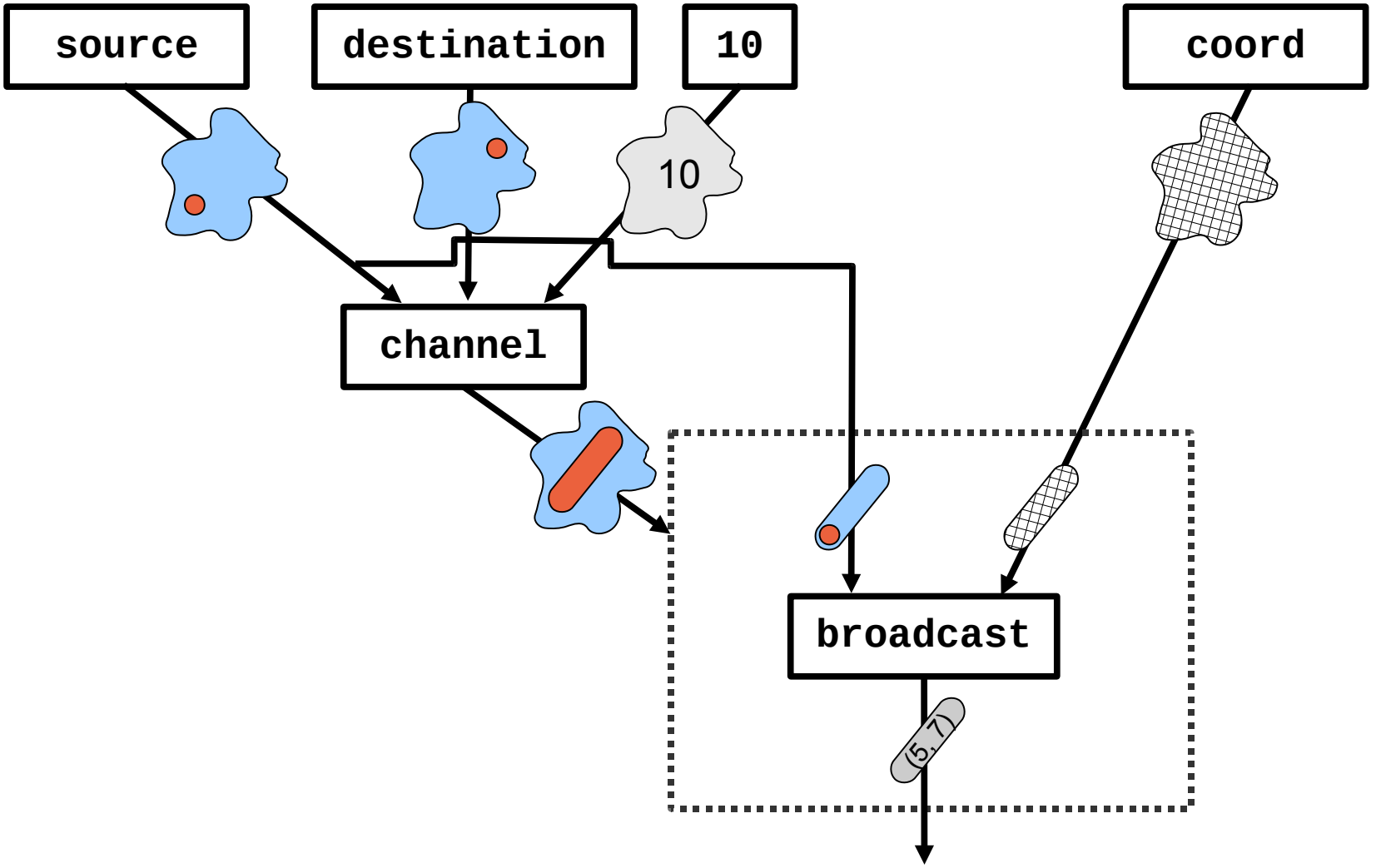


Feedback via Configuration Path

- **s(t+dt) = f(s(t),dt)**
  - Function can leap from value to value (unlike derivative, CT-feedback approaches)
  - Steps can be irregular
  - Smaller steps = better approximation

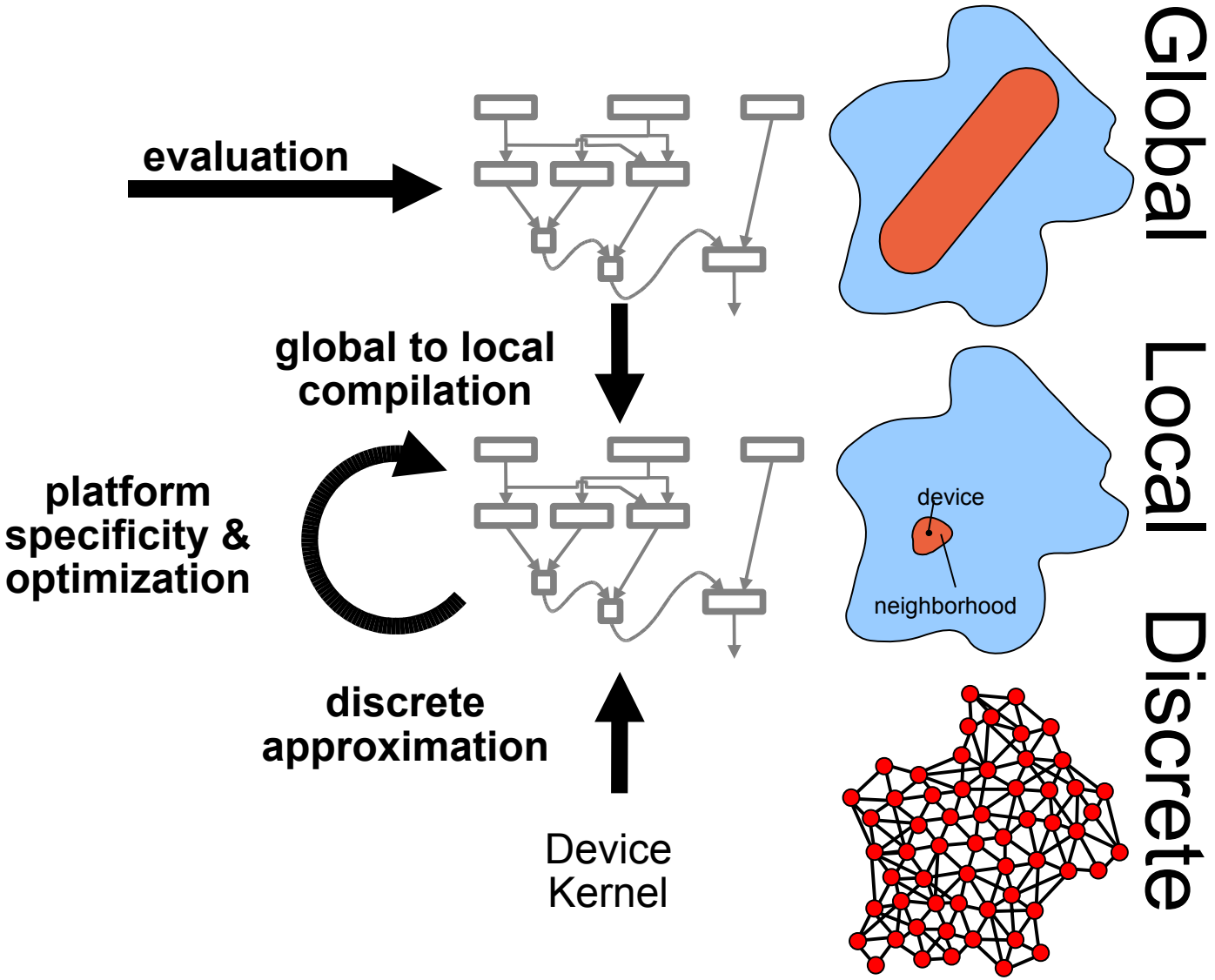


Modulation by Restriction



Proto

```
(def distance-to (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (distance-to src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
         (trail (<= (+ (distance-to src)
                       (distance-to dst))
                    d)))
    (dilate trail width)))
```



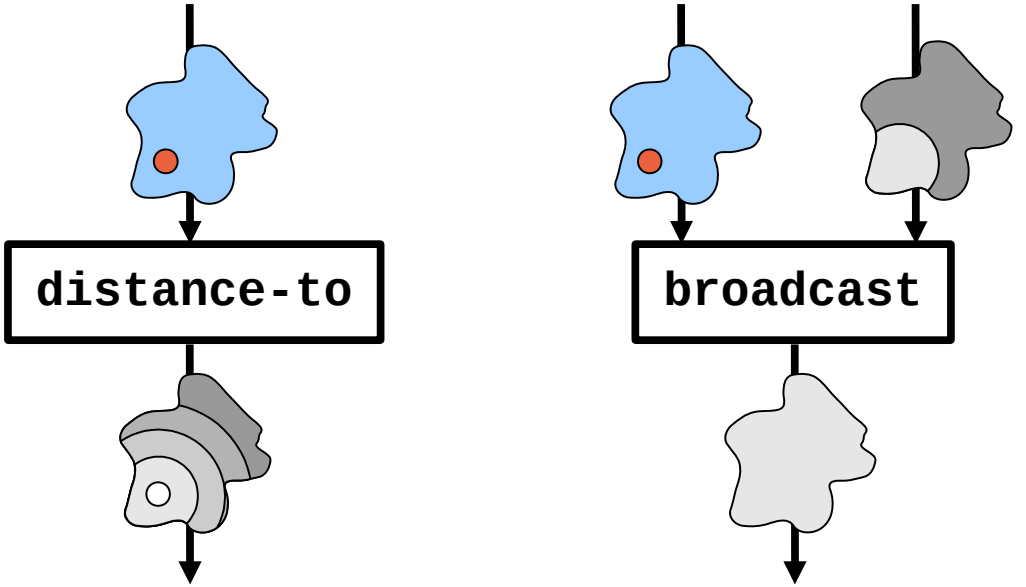
http://stpg.csail.mit.edu/proto.html

Weaknesses of Proto

- Functional programming scares people
- Programmers can break the abstraction
- No dynamic allocation of processes
- No formal proofs available for quality of approximation in a composed program

(active research on last two)

Implementing Geometric Primitives

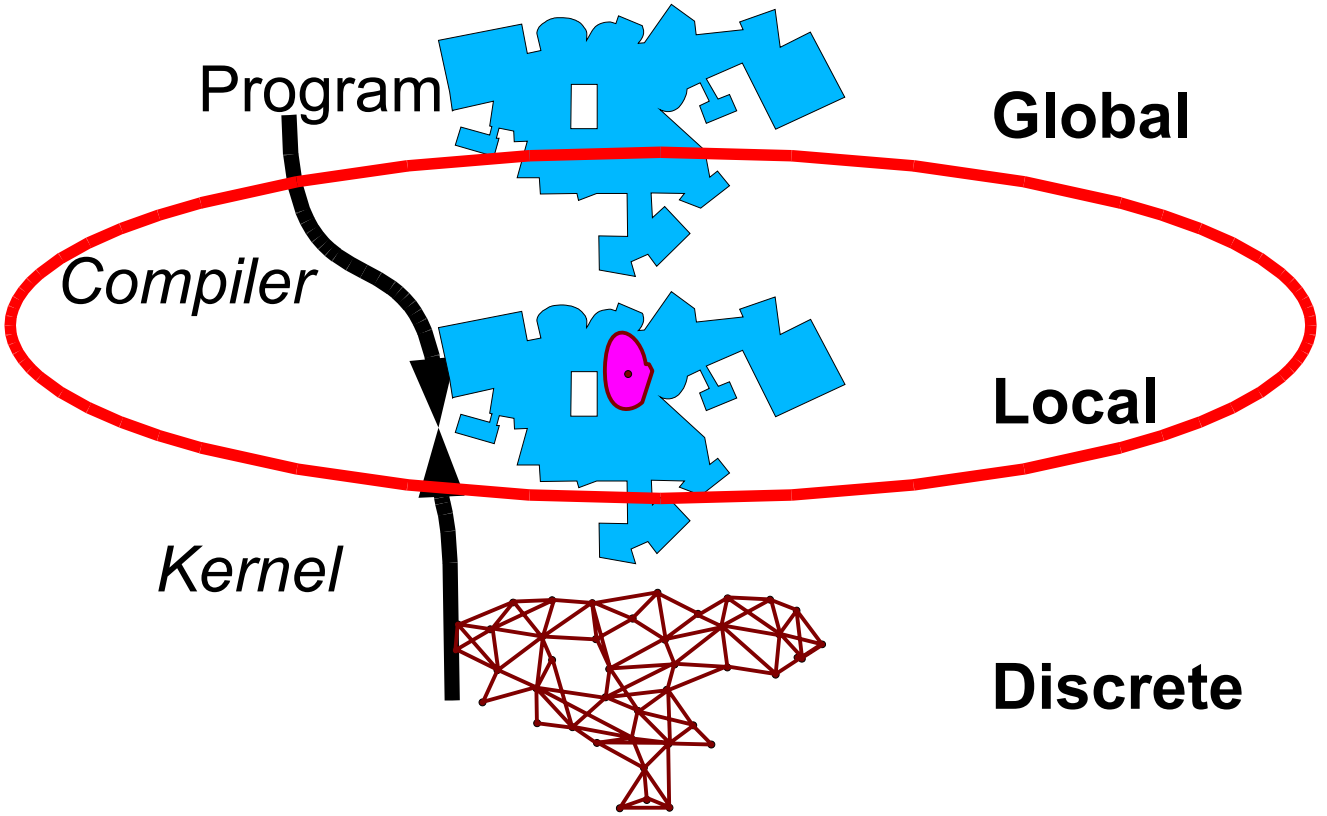


Let's go write the code for these...

Agenda

- Survey of Spatial Computing Approaches
- Amorphous Medium Abstraction
- Space-Time Primitives
- **Compiling from Global to Local**

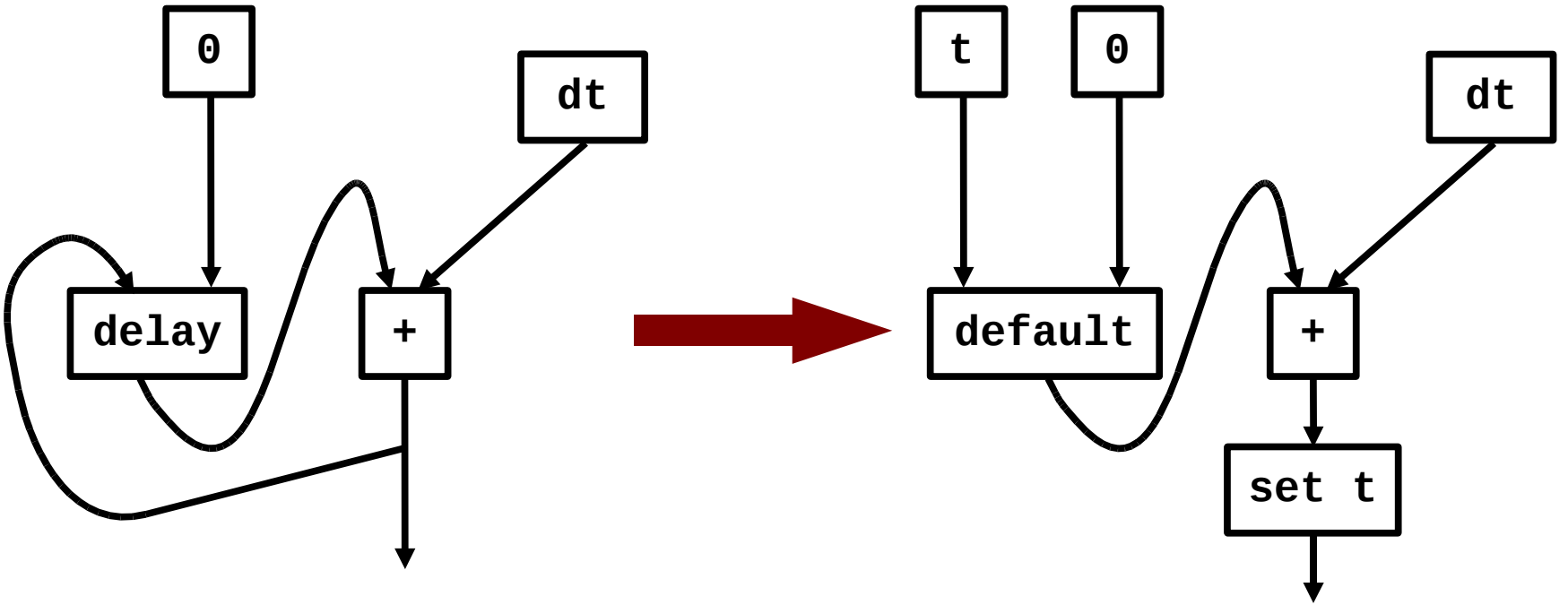
Global v. Local v. Discrete



Compiling from Global to Local

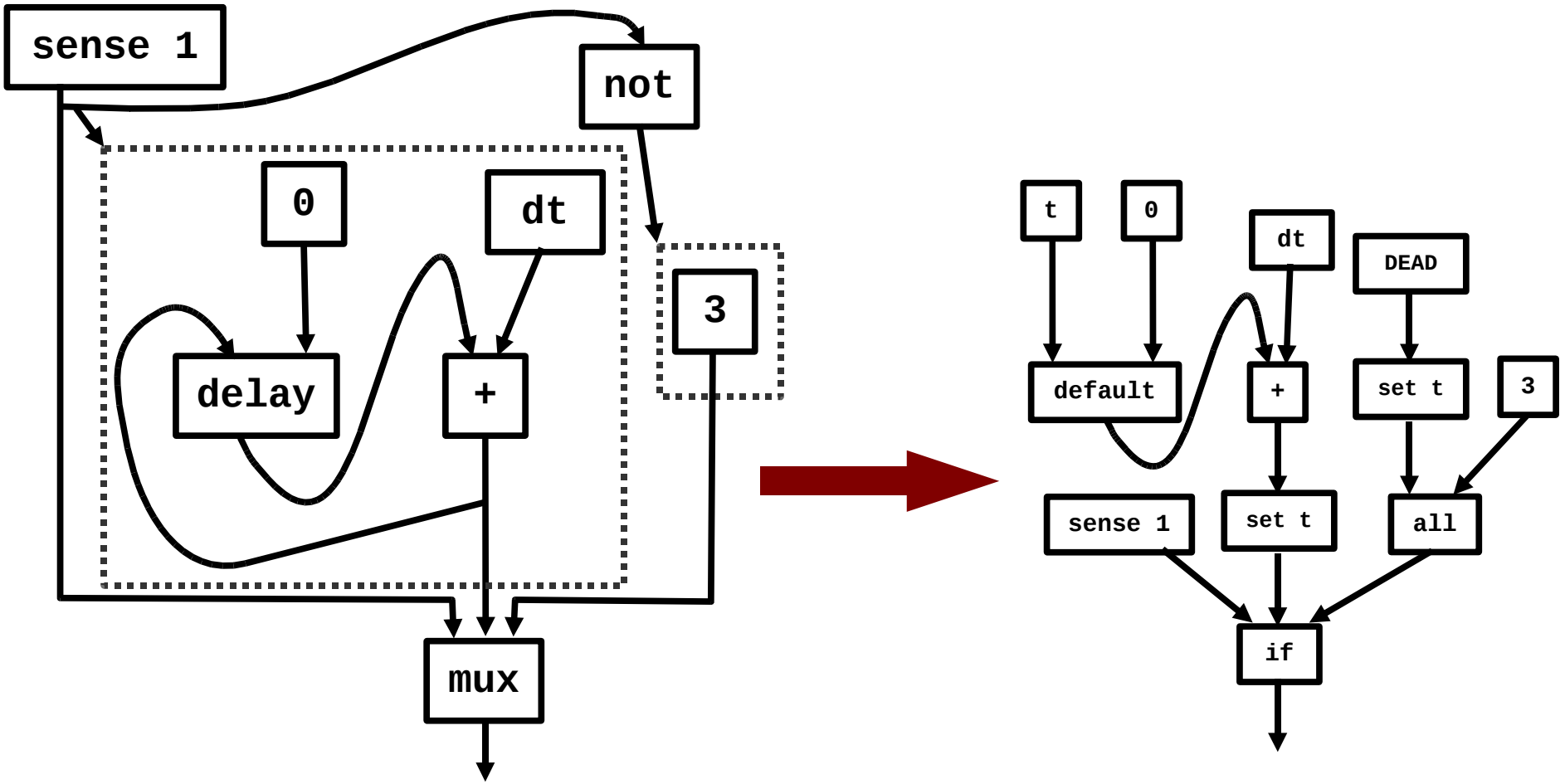
- Pointwise: no change
- Feedback → state update
- Restriction → branching & reset
- Neighborhood operations → per-neighbor computation
- Functional composition → tree walk & stack ops

Feedback → State Update



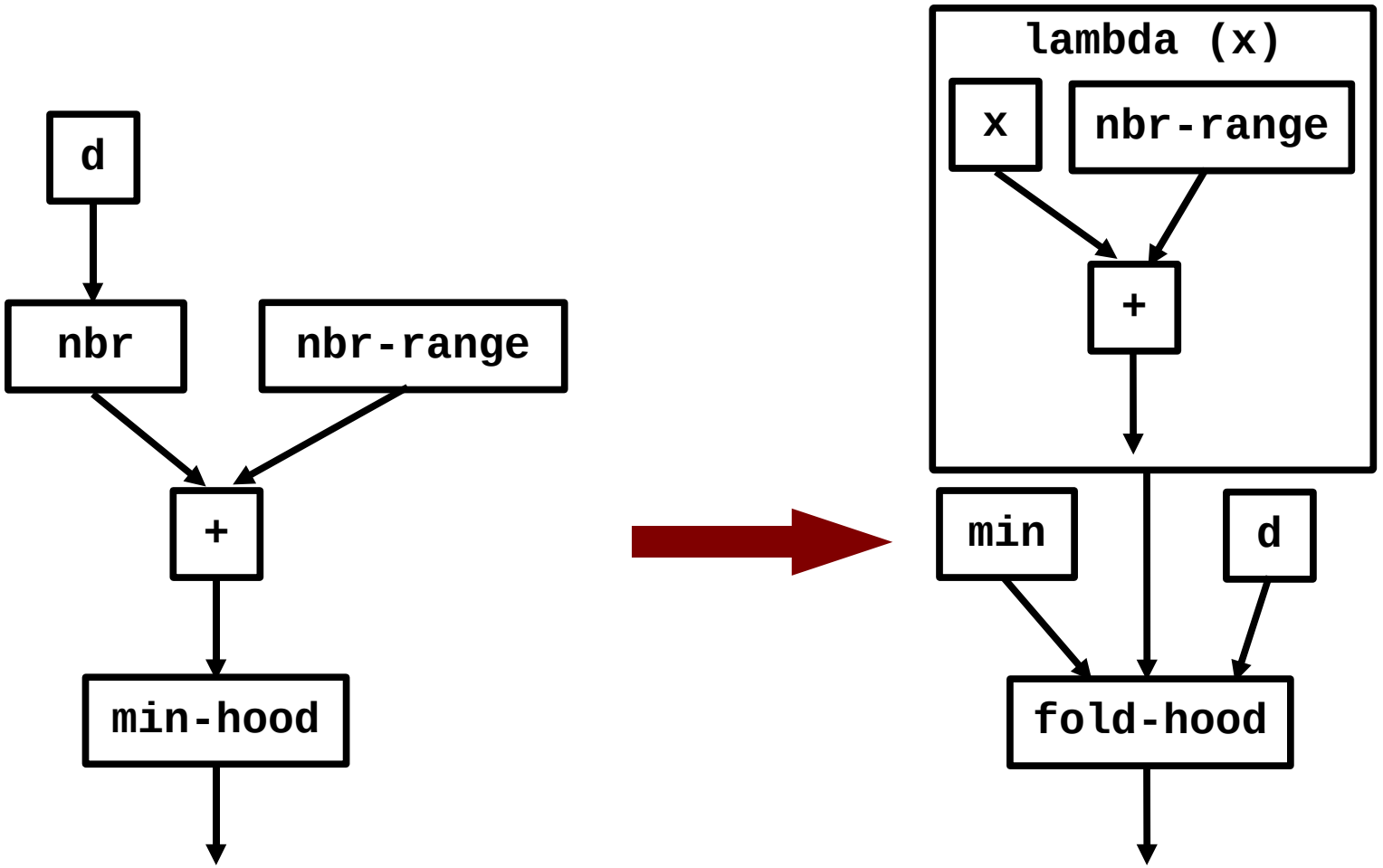

```
(rep t 0 (+ d (dt)))
```

Restriction → Branching & Reset



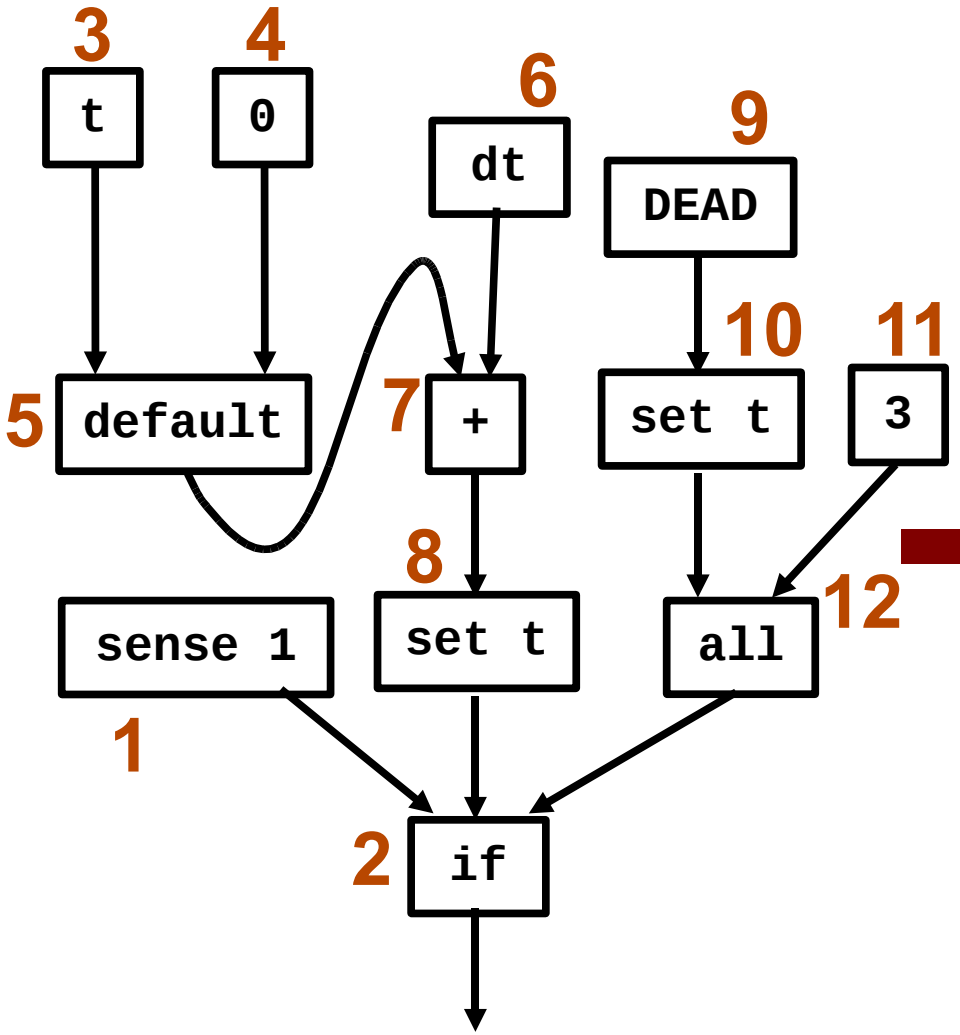

```
(if (sense 1) (rep t 0 (+ d (dt))) 3)
```

Neighborhood ops → Per-nbr computation




```
(min-hood (+ (nbr-range) (nbr d)))
```

Tree-walk linearizes to stack ops




```
00: SENSE 1
01: IF    09
02: REF 14
03: LIT 0
04: DEFAULT
05: DT
06: PLUS
07: SET 14
08: JMP 13
09: LIT DEAD
10: SET 14
11: POP
12: LIT 3
13: EXIT
14: <DEAD>
```

True branch

False branch

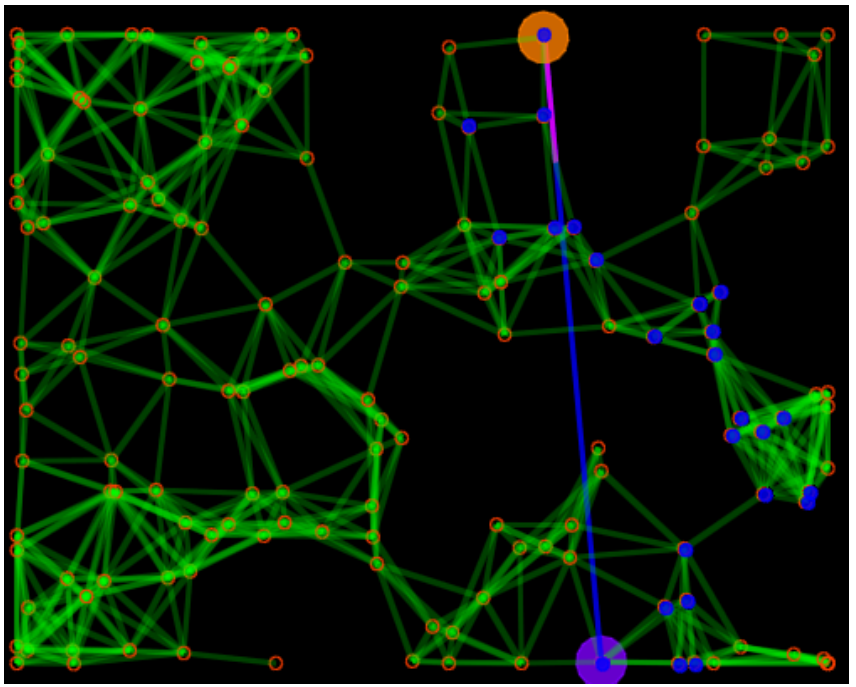
```
(if (sense 1) (rep t 0 (+ d (dt))) 3)
```

Most operators act last, branches act second

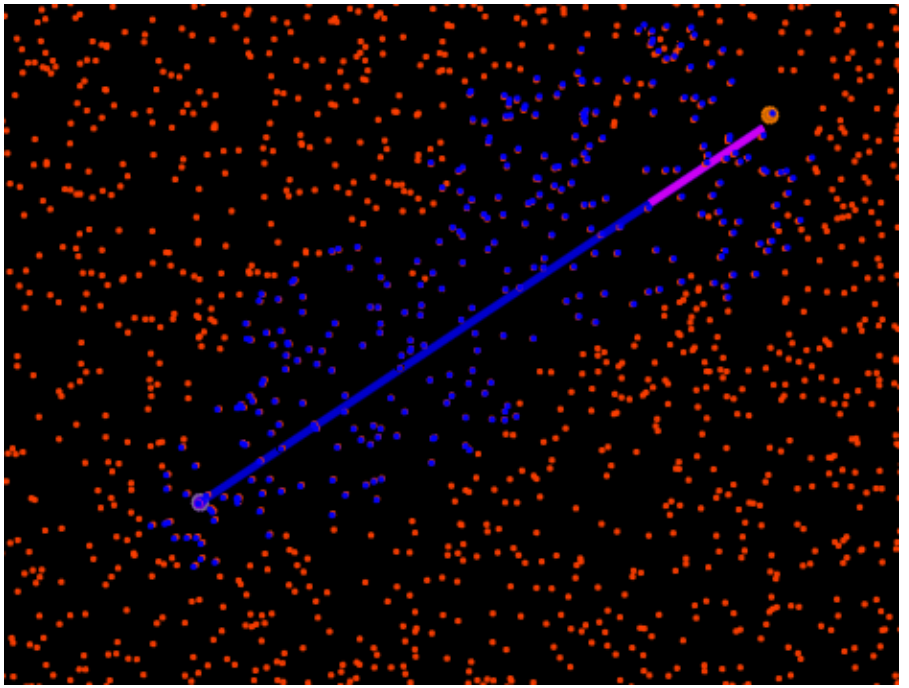
Summary

- Spatial programming approaches based on local dynamics give a good trade-off between expressiveness and abstraction.
- Amorphous Medium abstraction simplifies programming of space-filling networks
- Four families of space and time operations, make it easy to compile global programs into local actions on an amorphous medium
- Geometric metaphors allow complex spatial computing problems to be solved with very short programs.

Tomorrow: Discrete Approximation & Self-Healing



150 devices



2000 devices

Further Questions

- What sort of declarative programming can safely be transformed from global to local?
- Are there programs that cannot be expressed well in continuous space?