A Spatial Computing Approach to Distributed Algorithms

Jacob Beal

Asilomar Signals, Systems, & Computers
November, 2010
If the problem structure is geometric... take advantage of it!

… but can we deepen the mathematical foundations?
Outline

- What is Spatial Computing?
- Global → Local → Global
- From Space to Robustness & Scalability
Spatial Computers

Robot Swarms

Biological Computing

Sensor Networks

Reconfigurable Computing

Cells during Morphogenesis

Modular Robotics
Graphs

Crystalline (e.g. CAs)

Amorphous/Continuous

(w. Dan Yamins)
Graphs

Crystalline (e.g. CAs)

Amorphous/Continuous

(jitter)

(grain size)

(density)

(spatial complexity)
Graphs

Crystalline (e.g. CAs)

Amorphous/Continuous

spatial computing

density

space complexity

jitter

grain size

(w. Dan Yamins)
Space/Network Duality

How well does the network cover space?

What space is covered well by the network?
Example: Target Tracking
Example: Target Tracking

Intruder

Guard
Example: Target Tracking

Intruder

Guard
Programming Languages Need:

- 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
Outline

- What is Spatial Computing?
- **Global → Local → Global**
- From Space to Robustness & Scalability
Example: Target Tracking

Intruder

Guard
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

(c.f. Butera)
Geometric Program: Channel

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Computing with fields

distance-to source
distance-to destination
distance
+
<=
dilate
width
Computing with fields

source

distance-to

destination

distance-to

distance

+ 37

<=

dilate

width 10
Amorphous Medium

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

Approximate with:
- Discrete network of devices
- Signals transmit state
(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)))
Proto's Families of Primitives

Pointwise
- +
- Feedback
  - delay
  - +
  - 41
  - 7
  - 48

Restriction
- restrict
- Neighborhood
  - nbr
  - any-hood
In Simulation...
Outline

- What is Spatial Computing?
- Global → Local → Global
- From Space to Robustness & Scalability
Continuous Programs → Self-Scaling

Target tracking across three orders of magnitude
Robustness

- Local change adapts in discrete approximation
- Global change adapts in manifold geometry
Composition

- Purely functional → simpler composition
- Self-stabilizing geometric algorithms can be composed feed-forward
- Approximation error can be predicted
Weaknesses

- 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)*
Summary

- Amorphous Medium abstraction simplifies programming of space-filling networks
- Proto has four families of space and time operations, compiles global descriptions into local actions that approximate the global
- Geometric metaphors simplify the design of distributed algorithms that are scalable, adaptive, and robust.
Open Problems

• What is a method for computing approximation quality, given a primitive approximation model?

• What is a method for determining the critical space/time density for approximation failure?

• Is Proto space/time universal w.r.t. causal, finitely approximable computations?

• What are the lower bound efficiency costs of the continuous space abstraction?

  and many more...