

Spatial Computing, Synthetic Biology, and Emerging IP Challenges

Jacob Beal
November, 2010

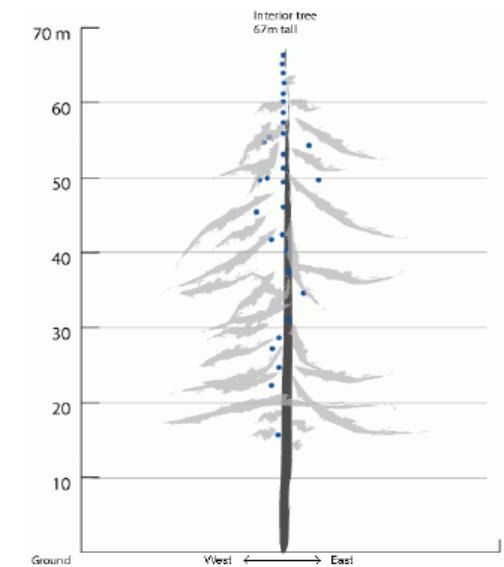
Spatial Computers



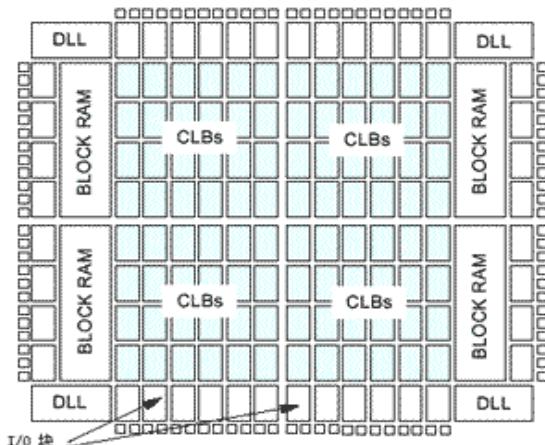
Robot Swarms



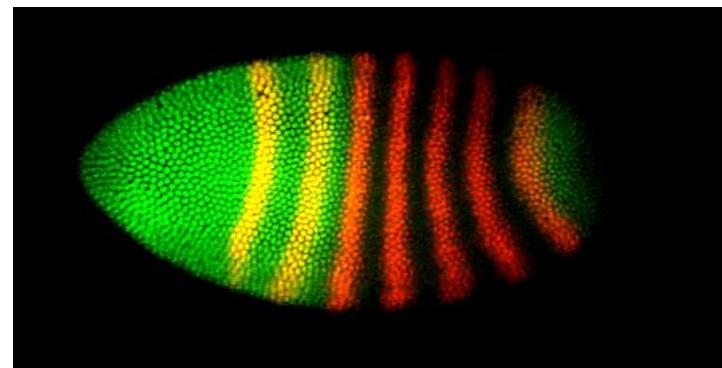
Biological Computing



Sensor Networks



Reconfigurable Computing



Cells during Morphogenesis



Modular Robotics

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

One answer: continuous space programs!

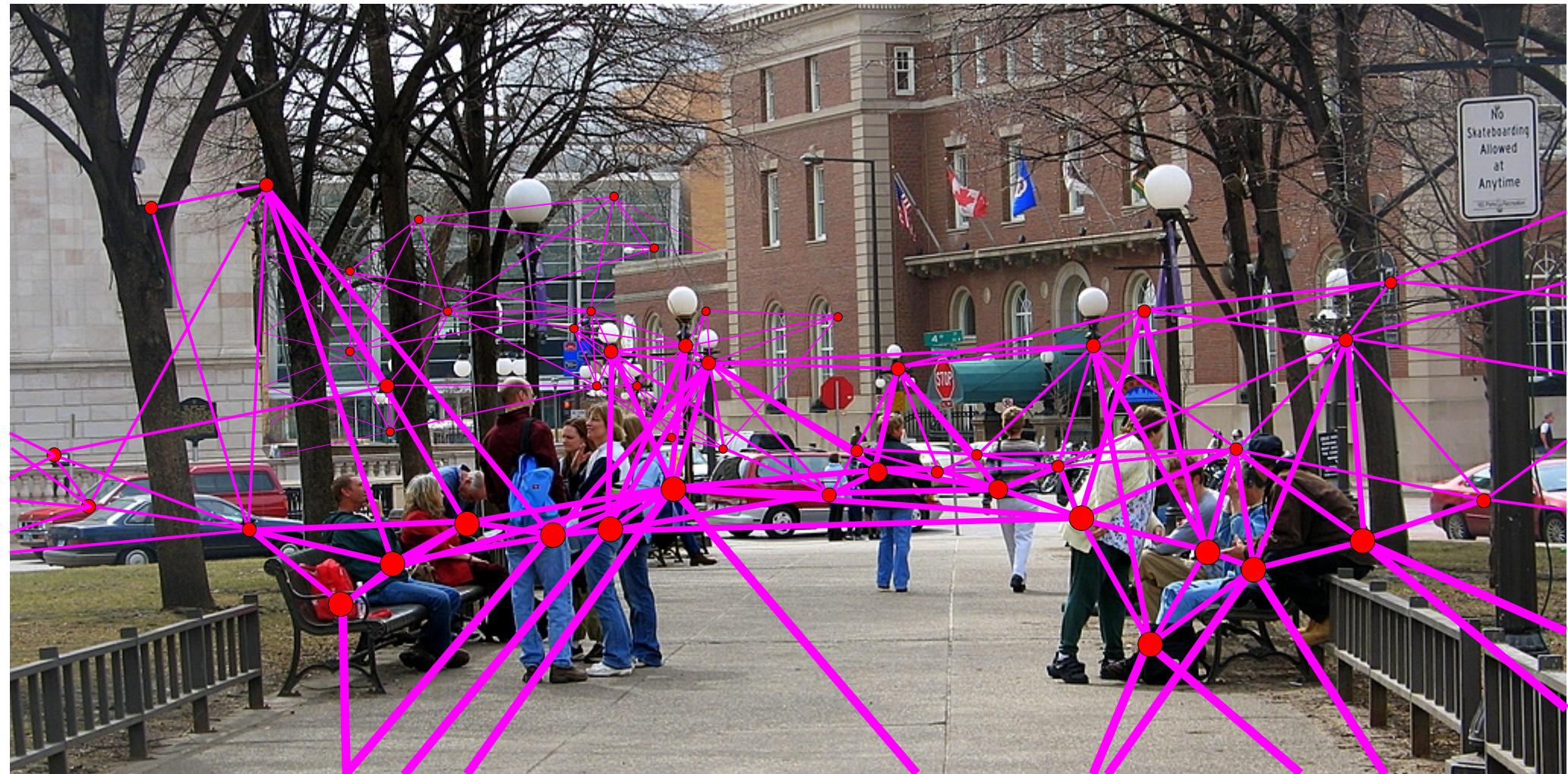
Example: Mobile Streaming



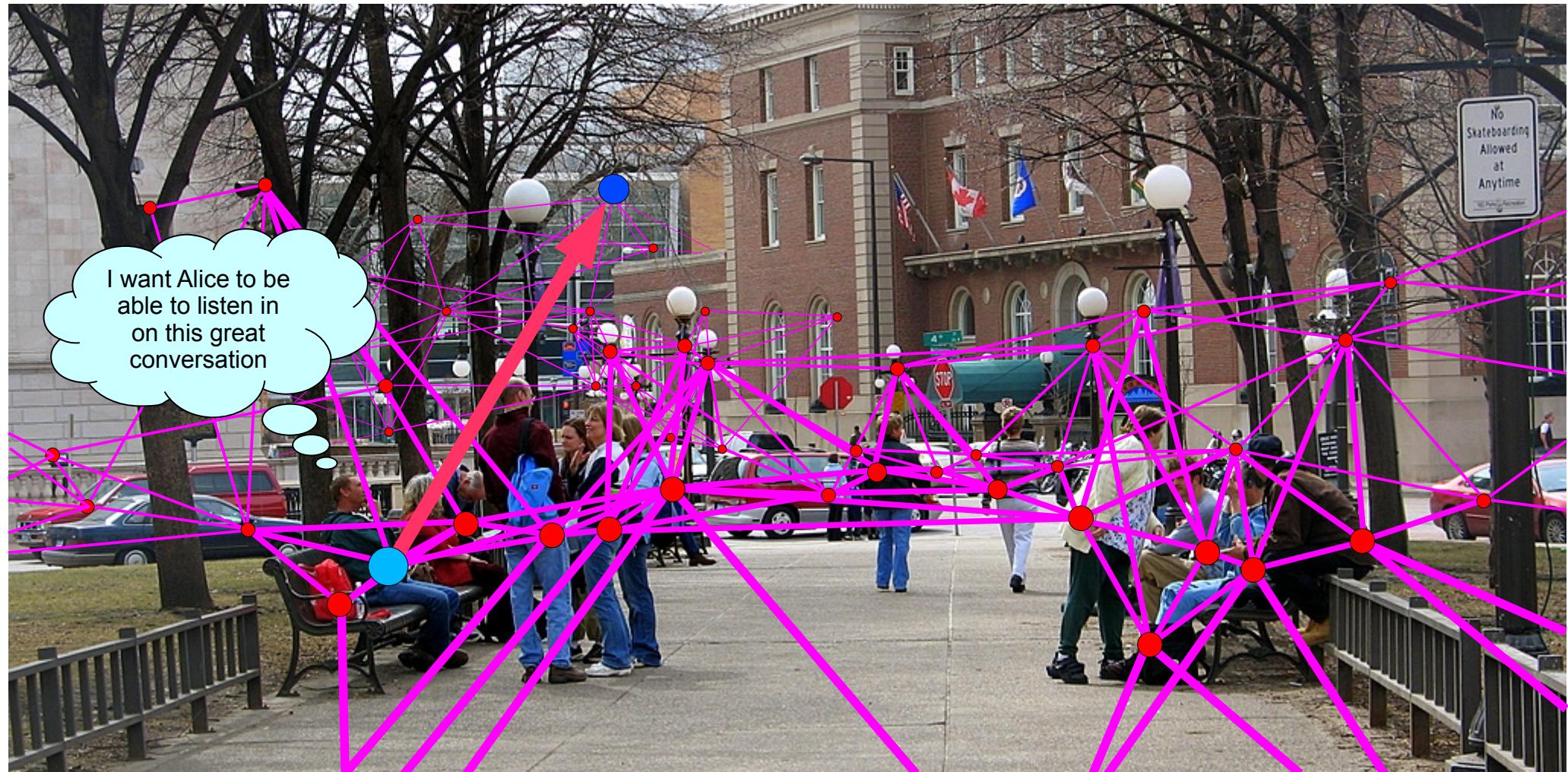
Example: Mobile Streaming



Example: Mobile Streaming



Example: Mobile Streaming

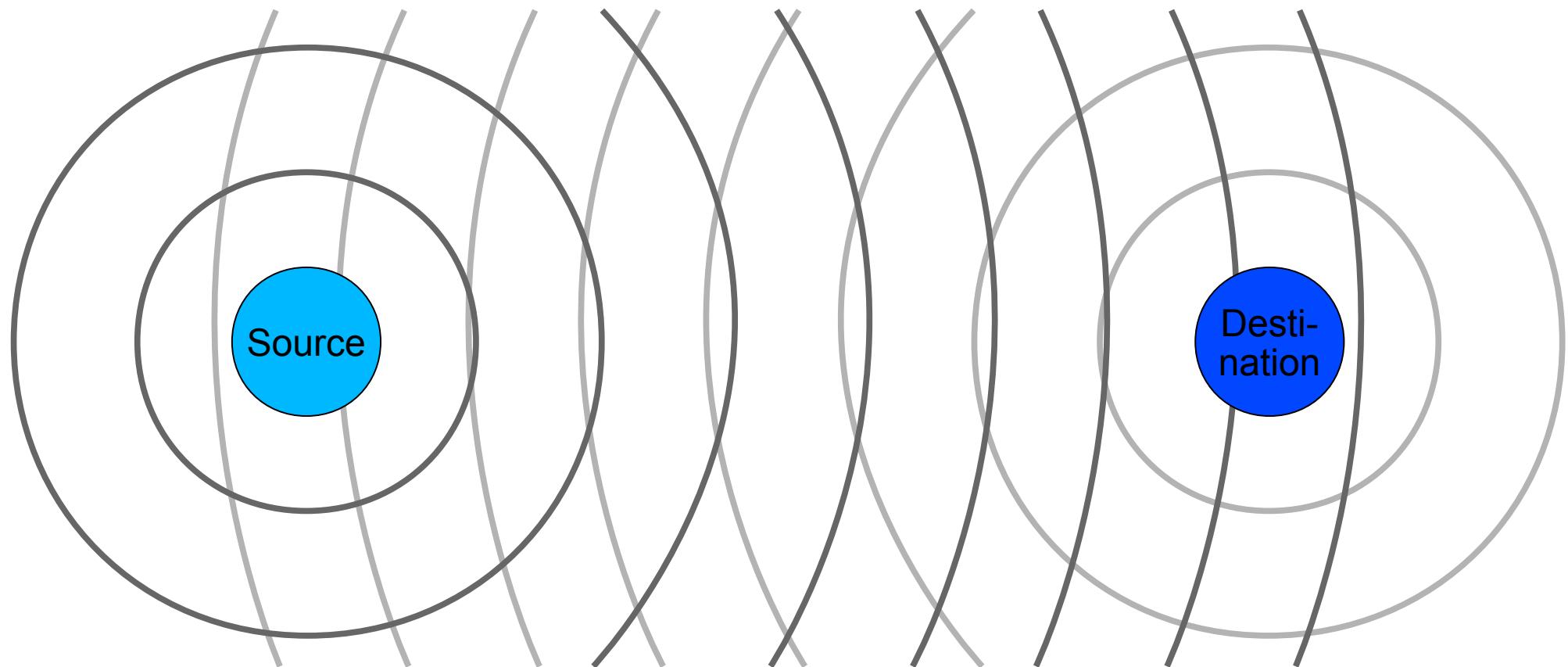


Geometric Program: Channel



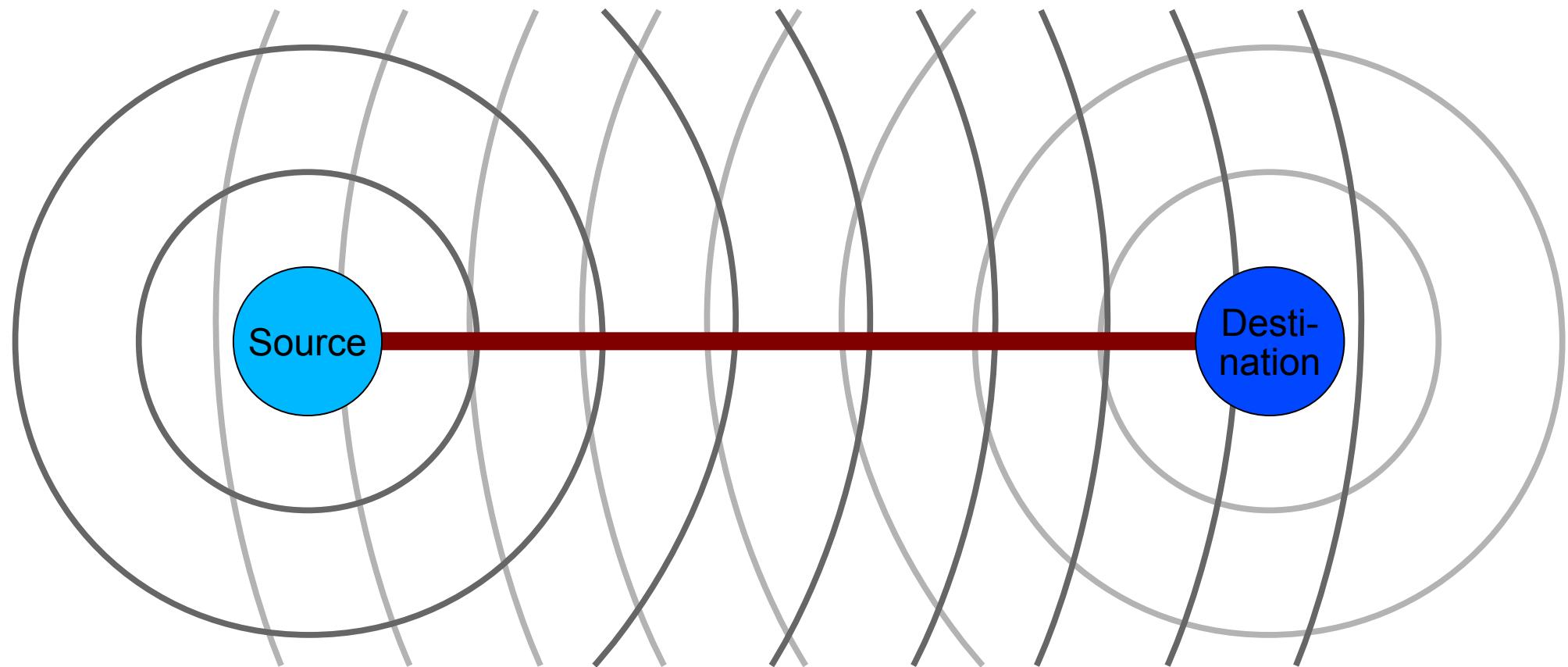
(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel



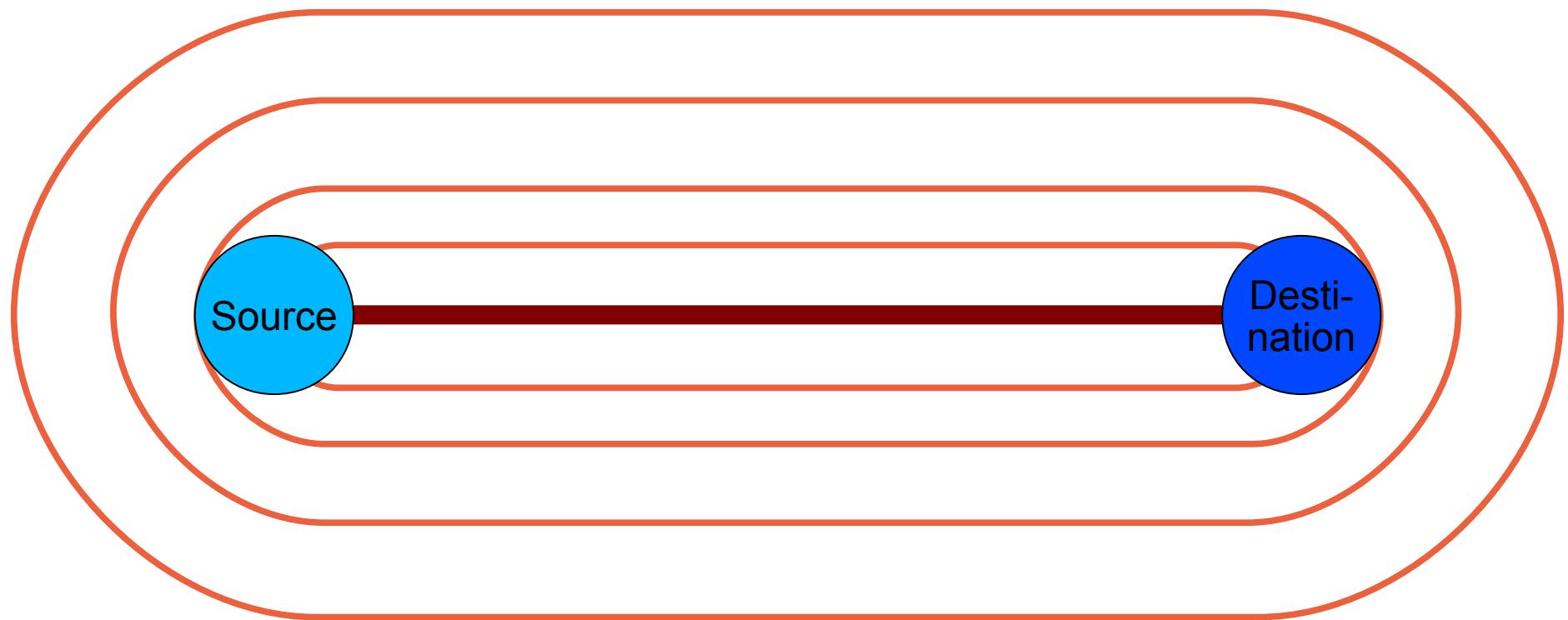
(cf. Butera)

Geometric Program: Channel



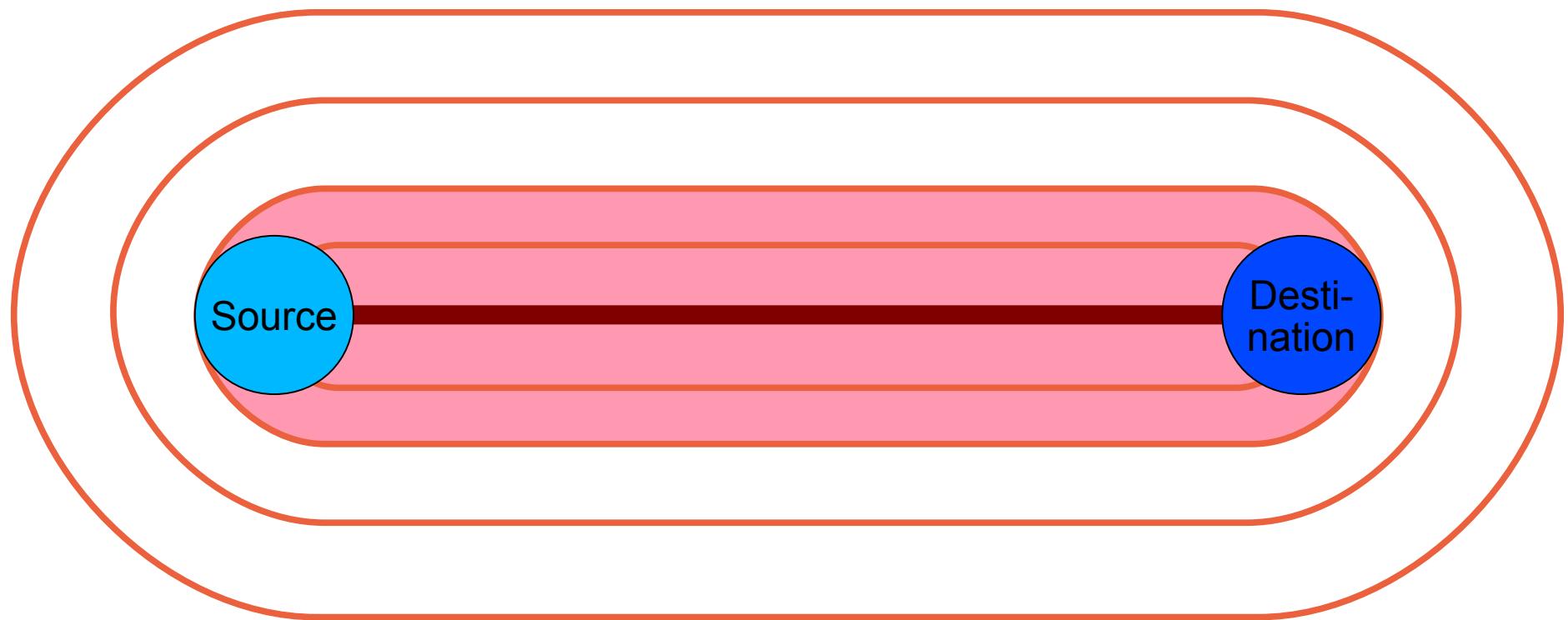
(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel



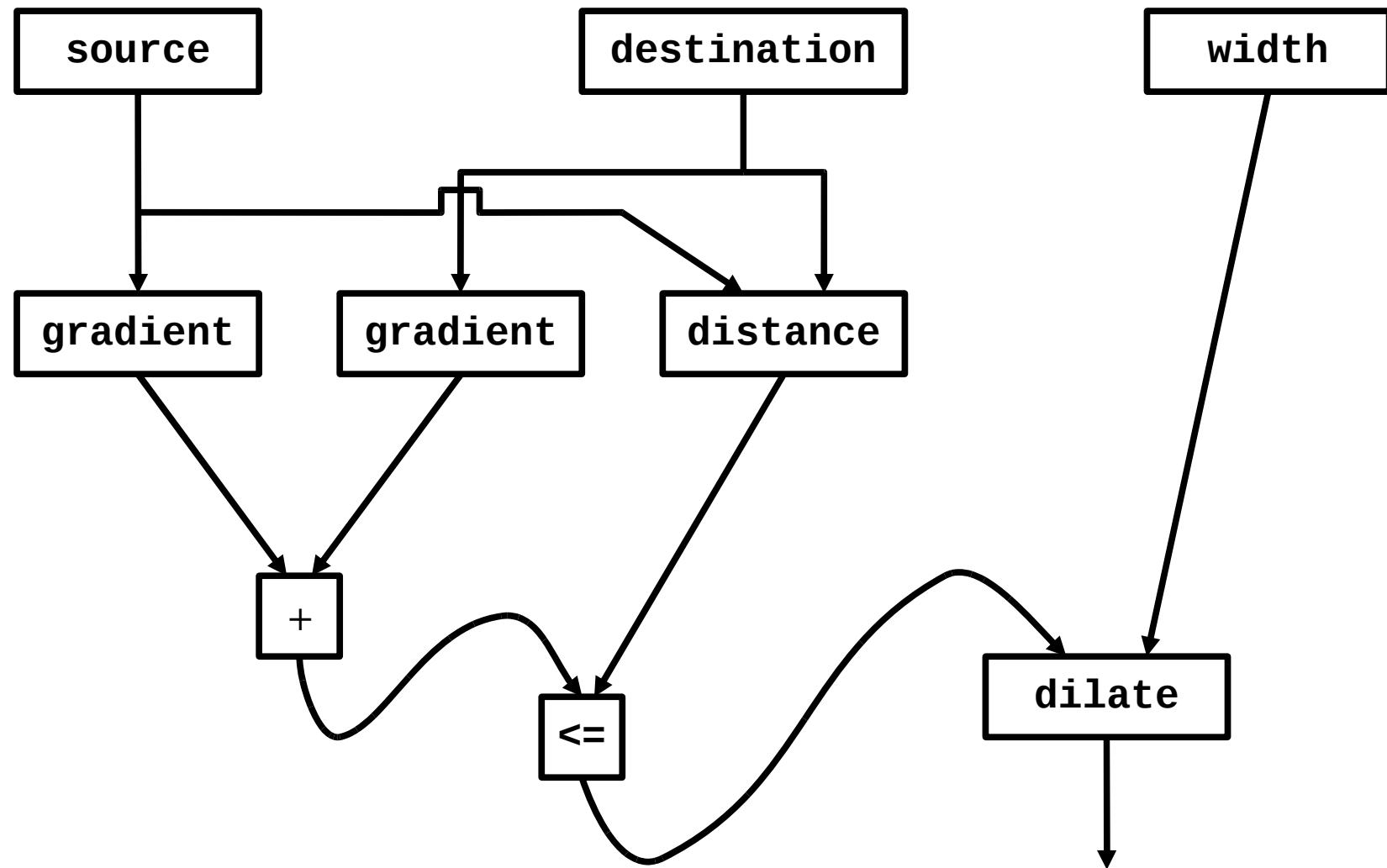
(cf. Butera)

Geometric Program: Channel

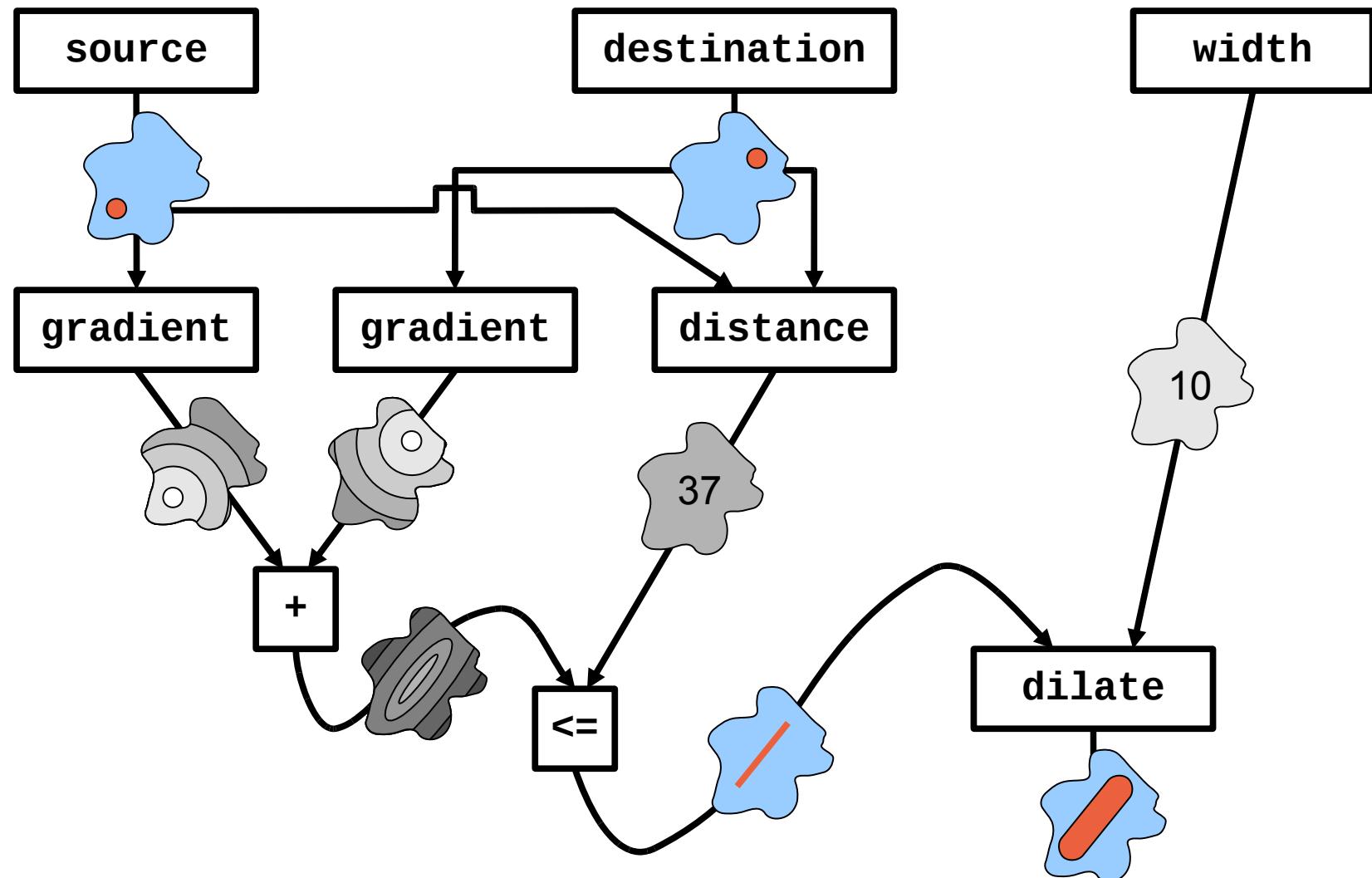


(cf. Butera)

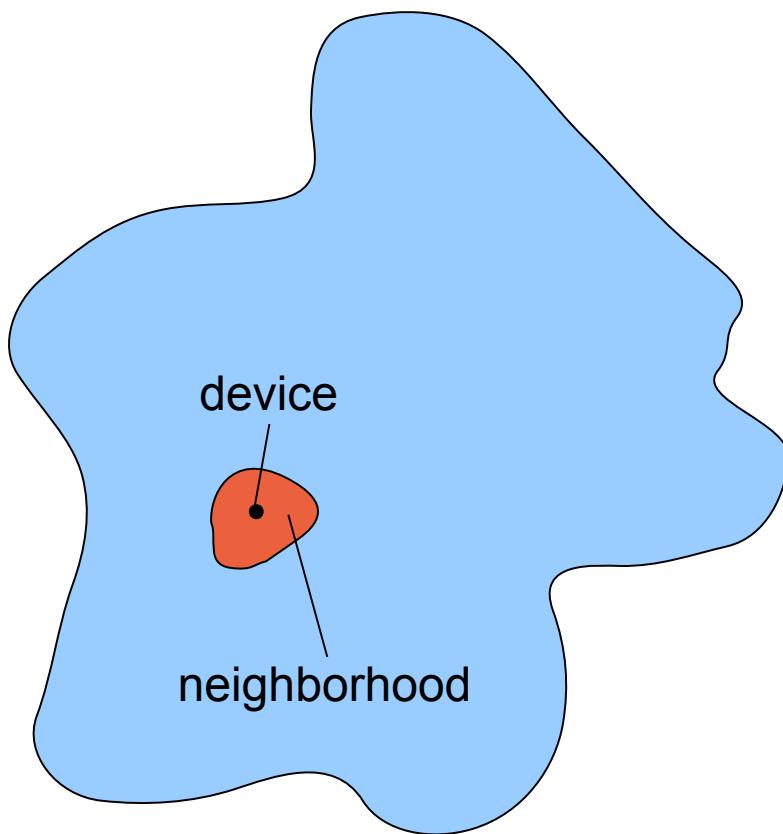
Computing with fields



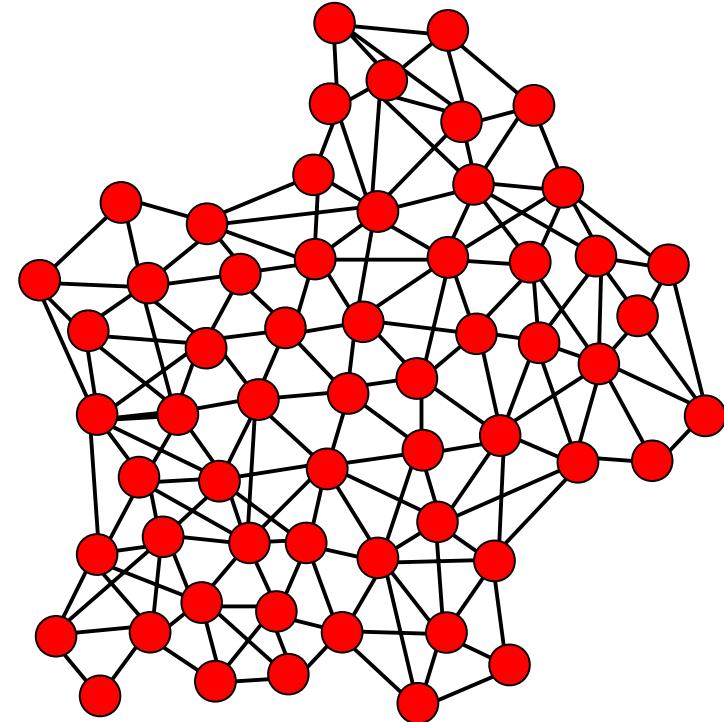
Computing with fields



Amorphous Medium



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



- Approximate with:
- Discrete network of devices
 - Signals transmit state

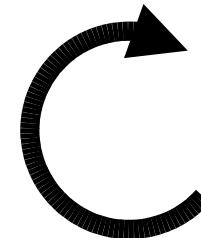
Proto

```
(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)))
```

**platform
specificity &
optimization**

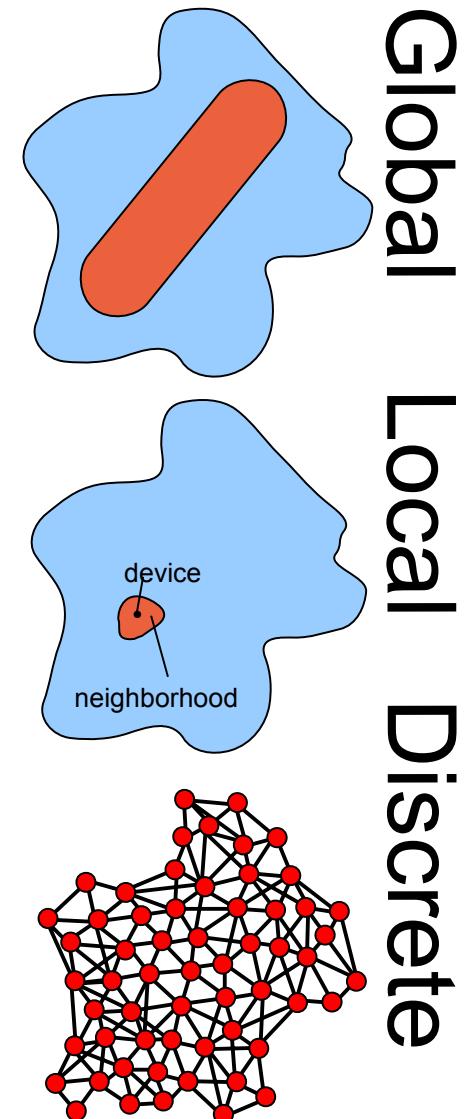
evaluation →

**global to local
compilation**



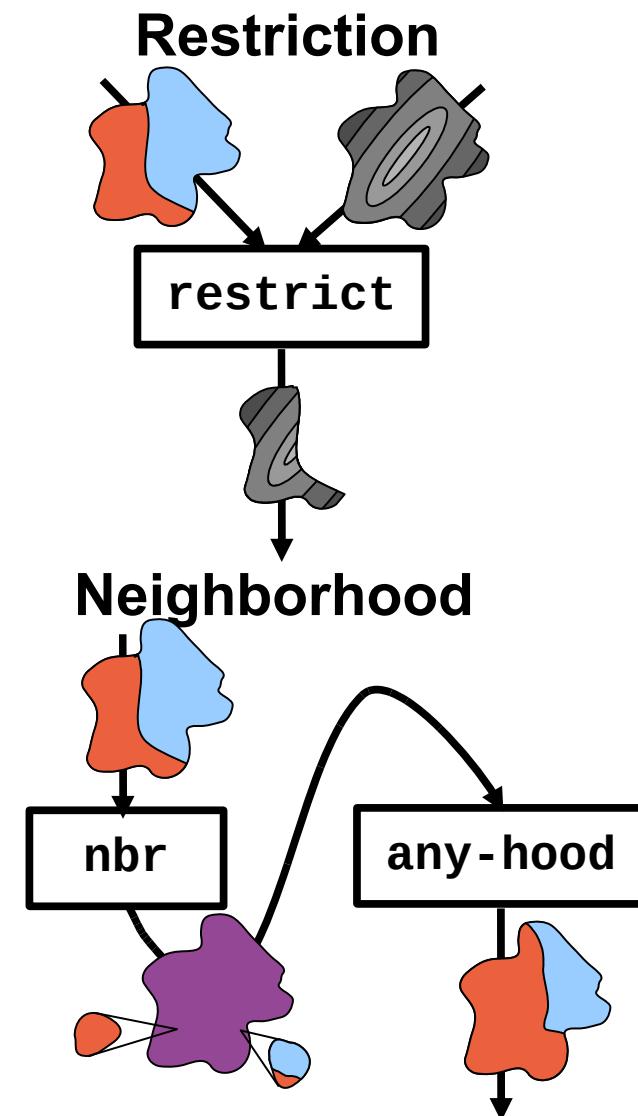
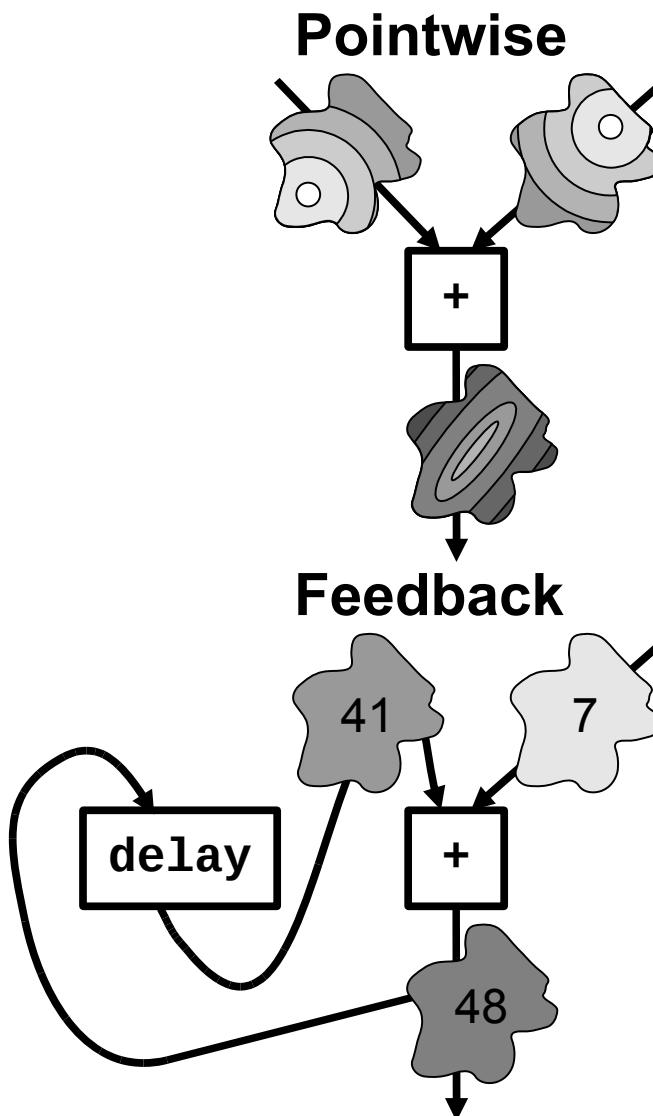
**discrete
approximation**

Device
Kernel



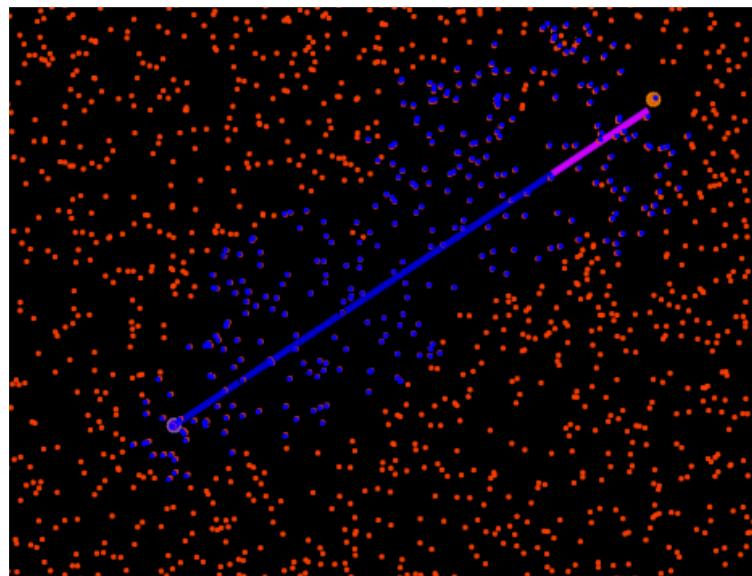
[Beal & Bachrach, '06]

Proto's Families of Primitives

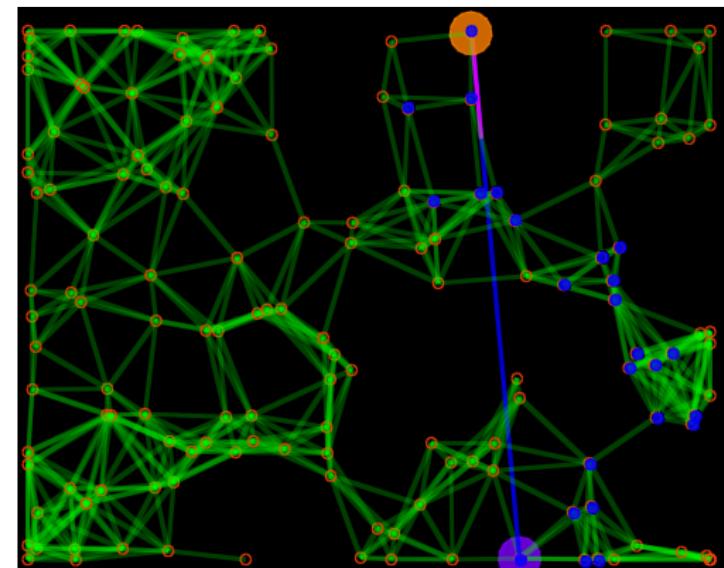


Why use continuous space?

- Scaling & Portability
- Robustness
- Composability

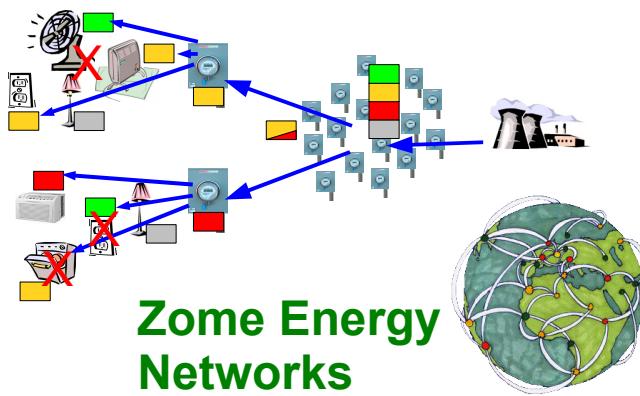


2000 devices

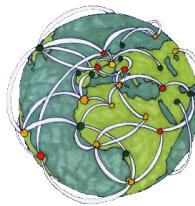


150 devices

Energy Management

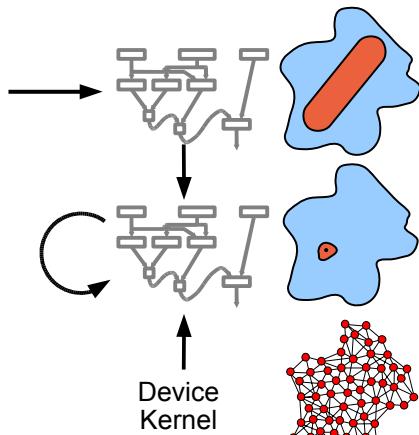


Zome Energy Networks



Proto

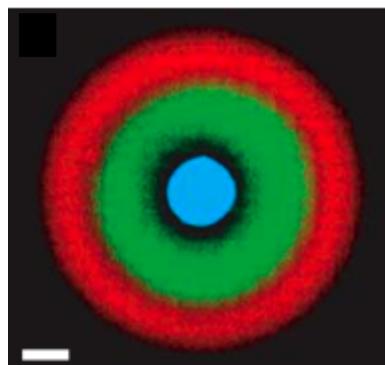
```
(def gradient (src ...)  
(def distance (src dst ...) ...)  
(def dilate (src n) ...  
(<= (gradient src n))  
(def channel (src dst width)  
(let* ((d (distance src dst))  
      (gradient dst))  
    (d)))  
(dilate trail width)))
```



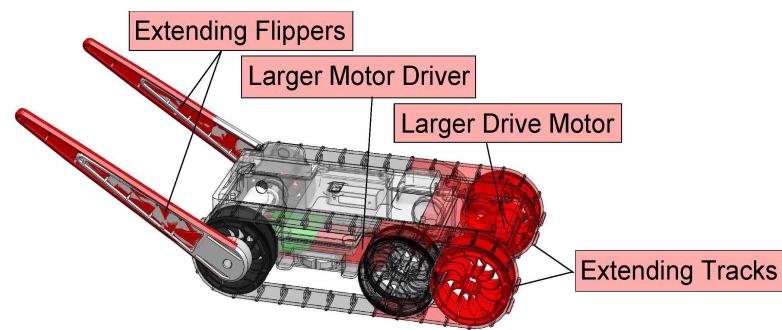
Swarm Robotics



Synthetic Biology

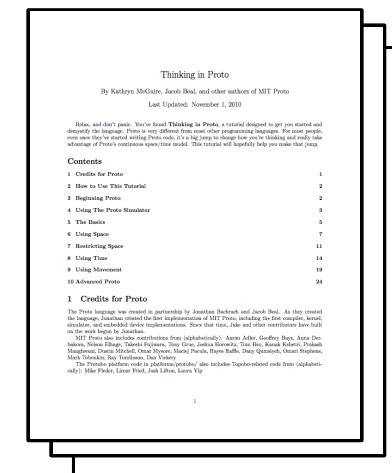
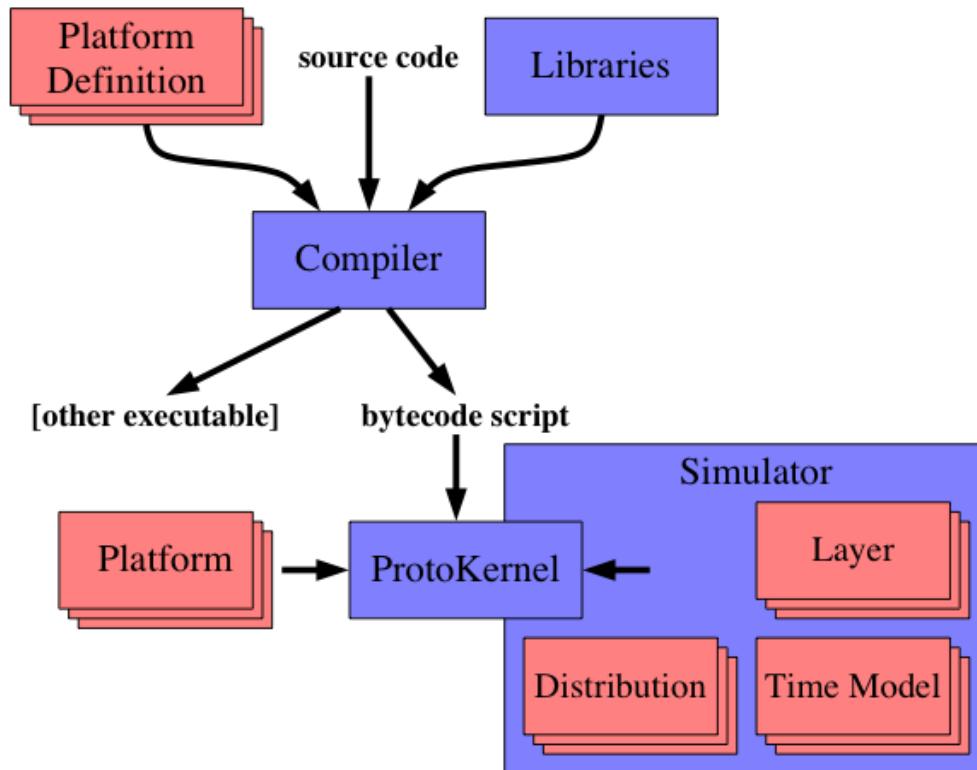


Morphogenetic Engineering



IP Challenges

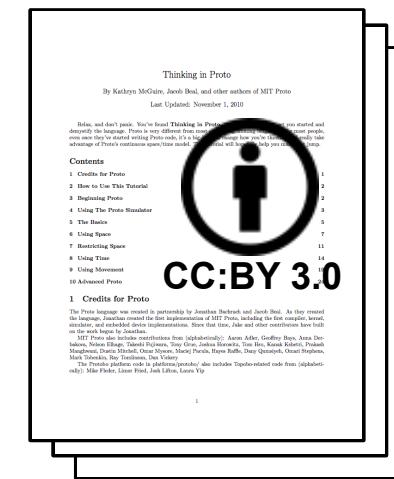
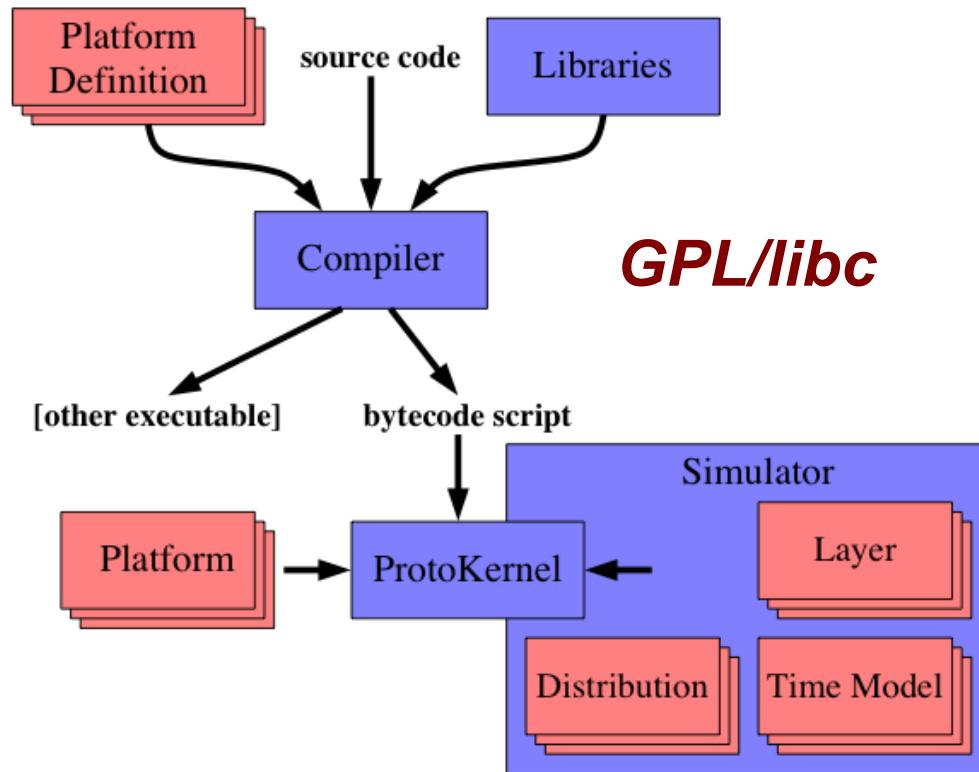
Many parts: free, protected, & commercializable?



Documentation

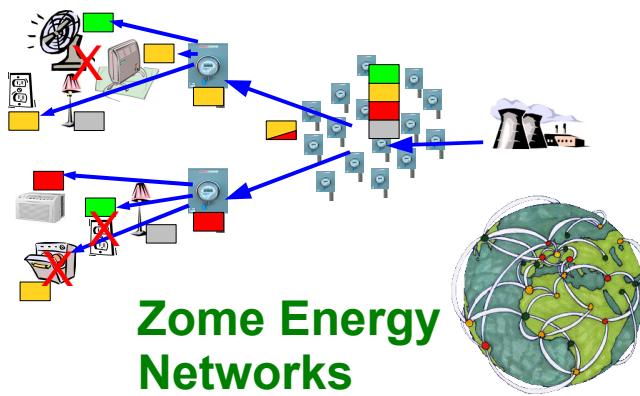
IP Challenges

Many parts: free, protected, & commercializable?

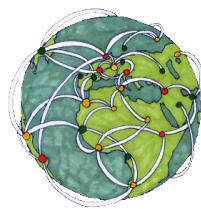


Thank you, Creative Commons!

Energy Management

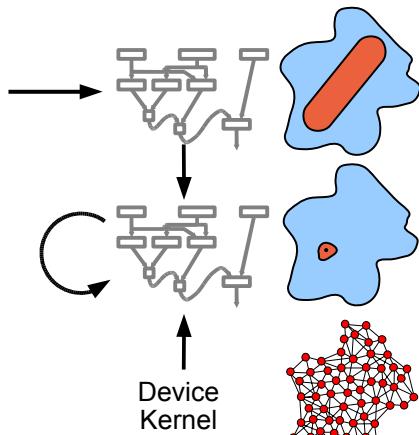


Zome Energy Networks



Proto

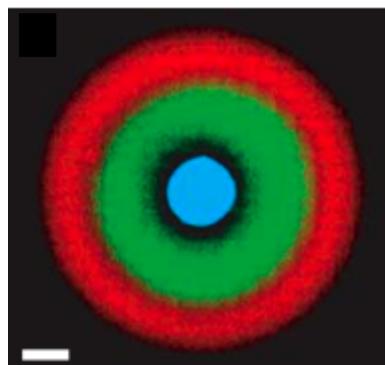
```
(def gradient (src ...)  
(def distance (src dst ...) ...)  
(def dilate (src n) ...  
(<= (gradient src n))  
(def channel (src dst width)  
(let* ((d (distance src dst))  
      (gradient dst))  
    (d)))  
(dilate trail width)))
```



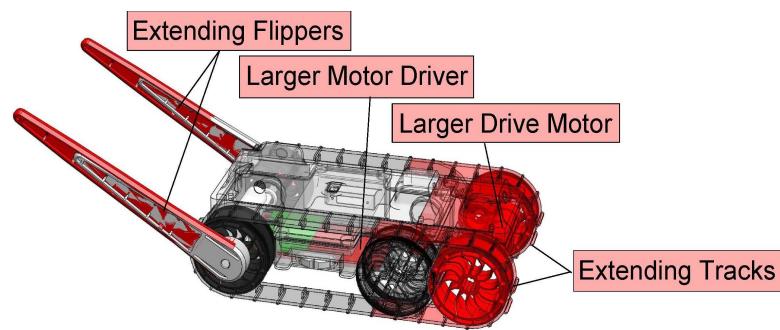
Swarm Robotics



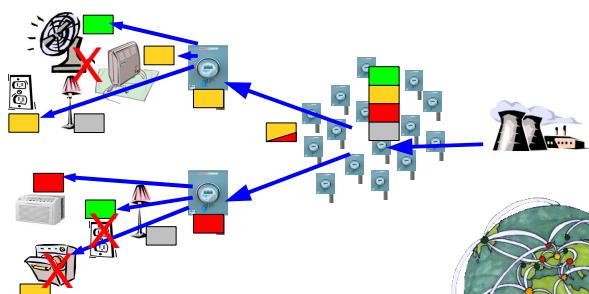
Synthetic Biology



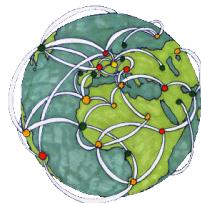
Morphogenetic Engineering



Energy Management

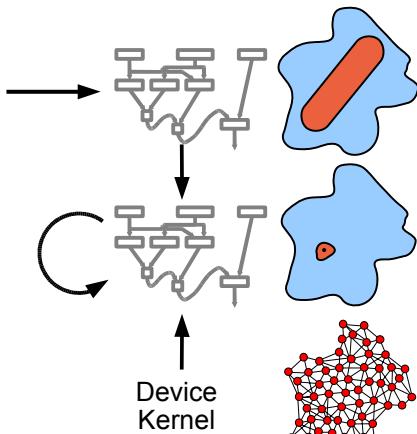


Zome Energy Networks

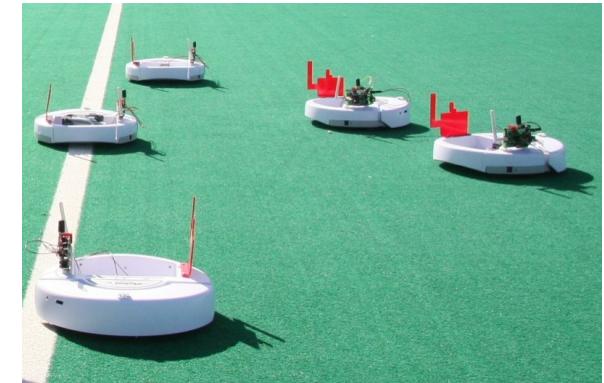


Proto

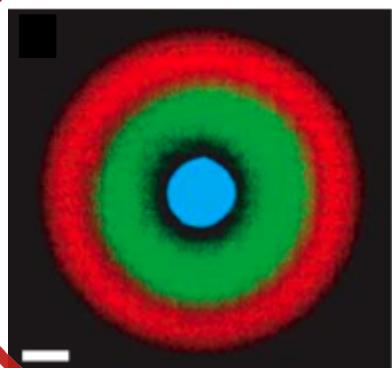
```
(def gradient (src ...)  
(def distance (src dst ...) ...)  
(def dilate (src n))  
(<= (gradient src n))  
(def channel (src dst width)  
(let* ((d (distance src dst))  
      (gradient dst))  
    d)))  
(dilate trail width)))
```



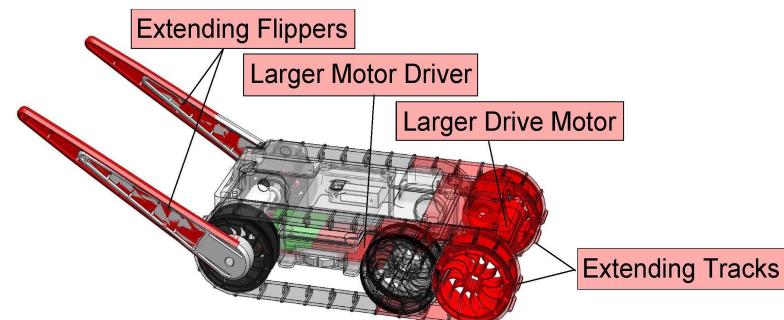
Swarm Robotics



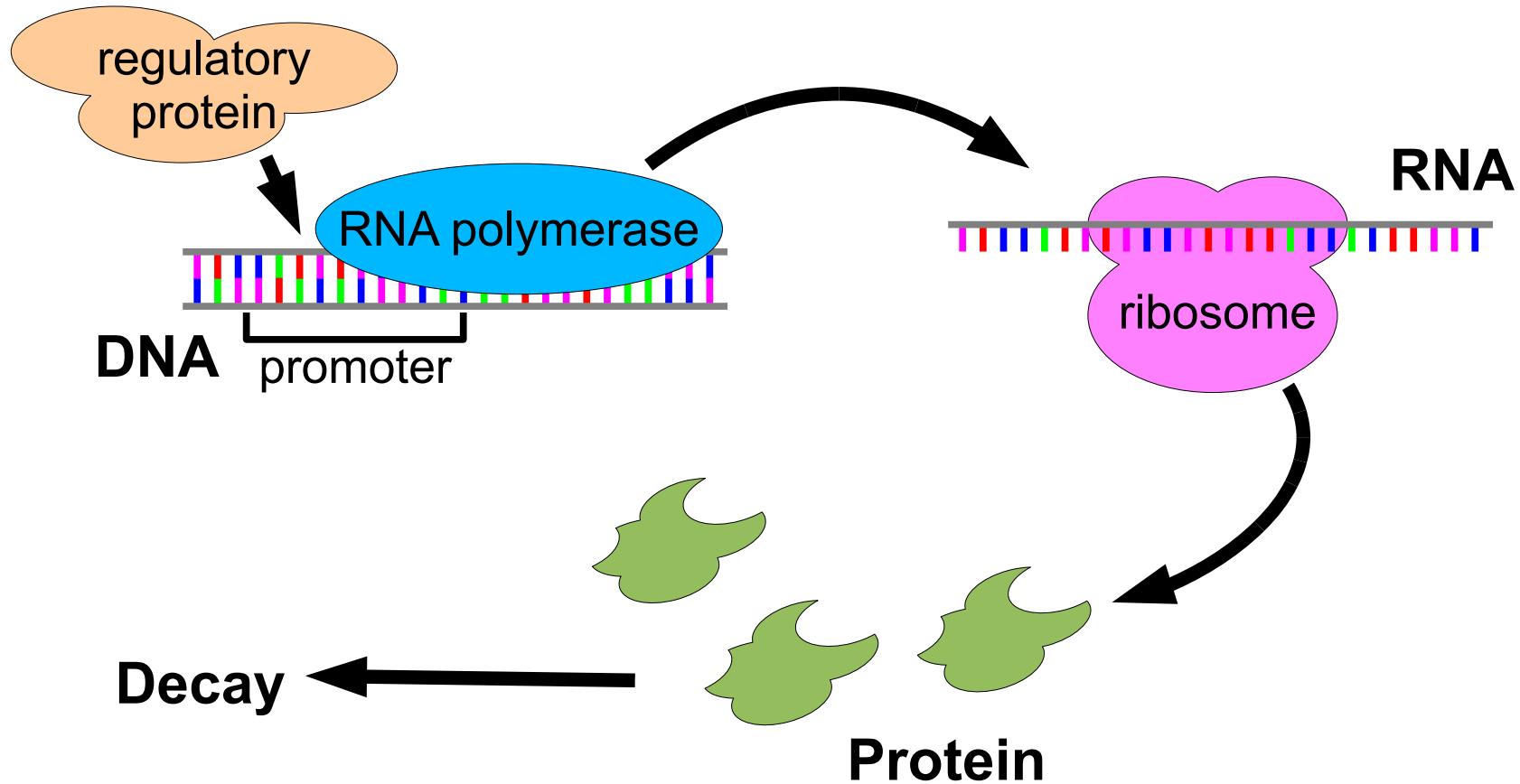
Synthetic Biology



Morphogenetic Engineering



Computation via Transcription Network



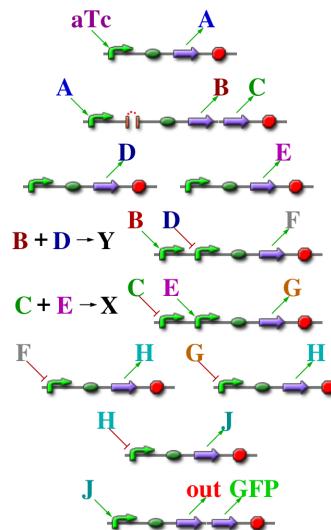
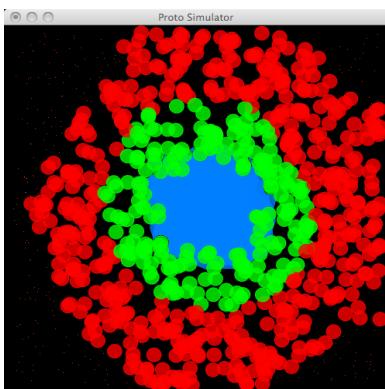
Proto BioCompiler

High-Level Language

```
(def band-detector (signal lo hi)
  (and (> signal lo)
       (< signal hi)))
(let ((v (diffuse (aTc) 0.8 0.05)))
  (green (band-detect v 0.2 1)))
```

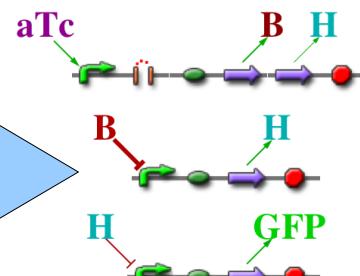
Compile

Simulate

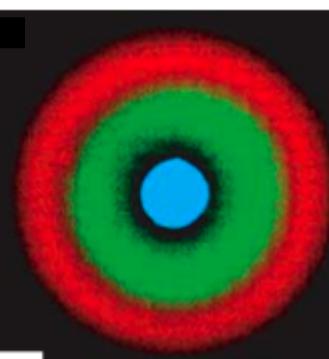


Genetic Regulatory Network

Optimize



Assemble



w. Weiss

Band detect: code

Proto

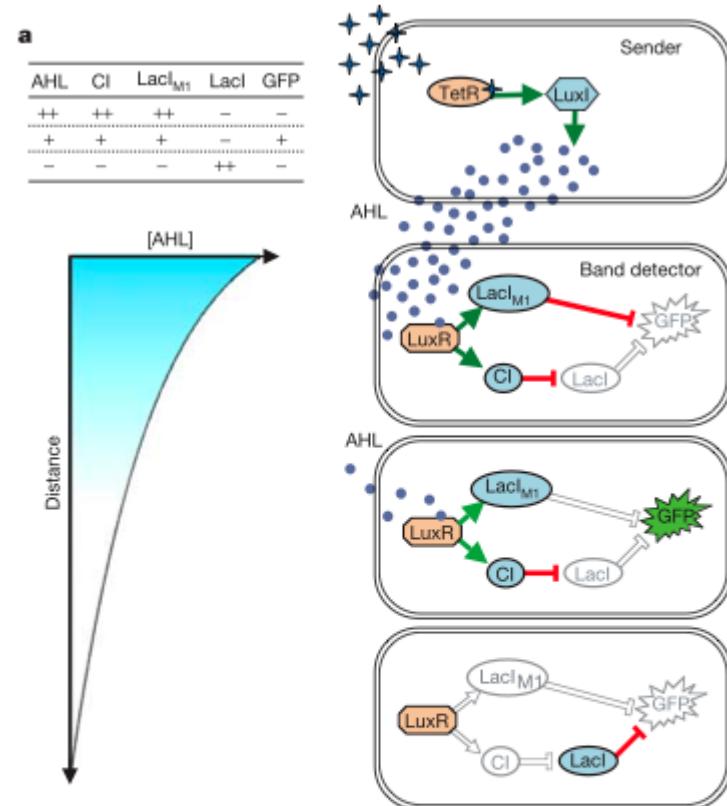
```
(def band-detector (signal lo hi)
  (and (> signal lo)
        (< signal hi)))
```

```
(let
  ((v (diffuse (aTc) 0.8 0.05)))
  (green (band-detect v 0.2 1)))
```

simpler, more reusable

[Beal & Bachrach, '08]

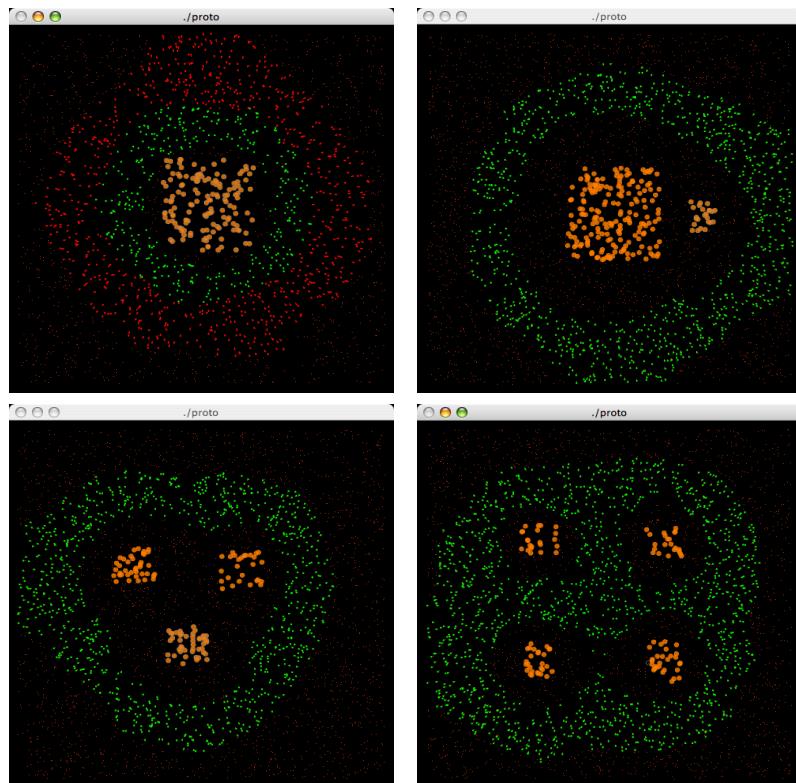
Engineered Bacteria



[Weiss '05]

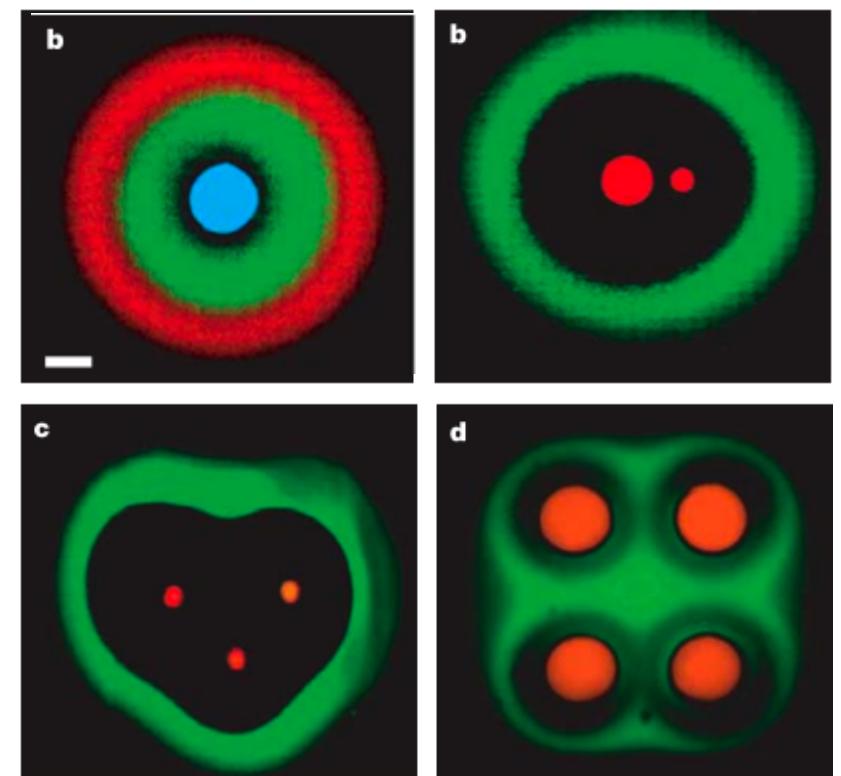
Band detect: behavior

Proto



[Beal & Bachrach, '08]

Engineered Bacteria



[Weiss '05]

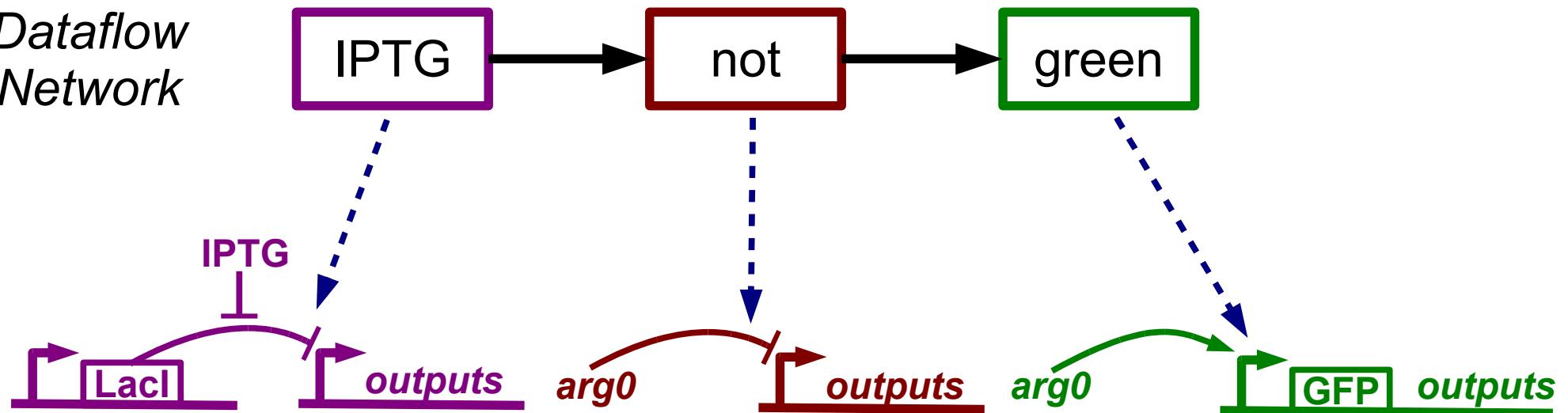
Motif-based Compilation

*Dataflow
Network*



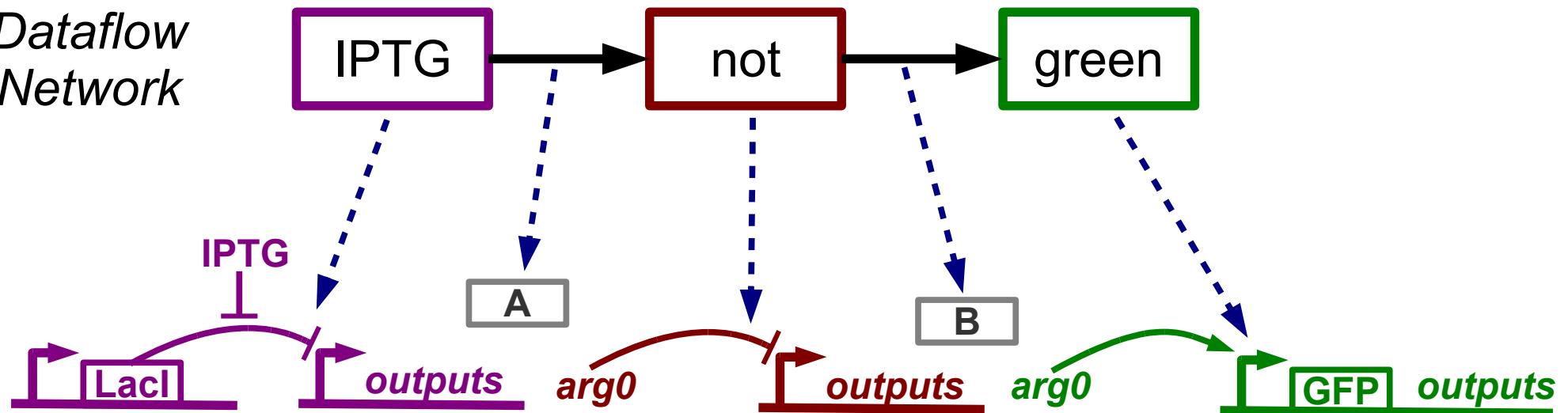
Motif-based Compilation

Dataflow Network

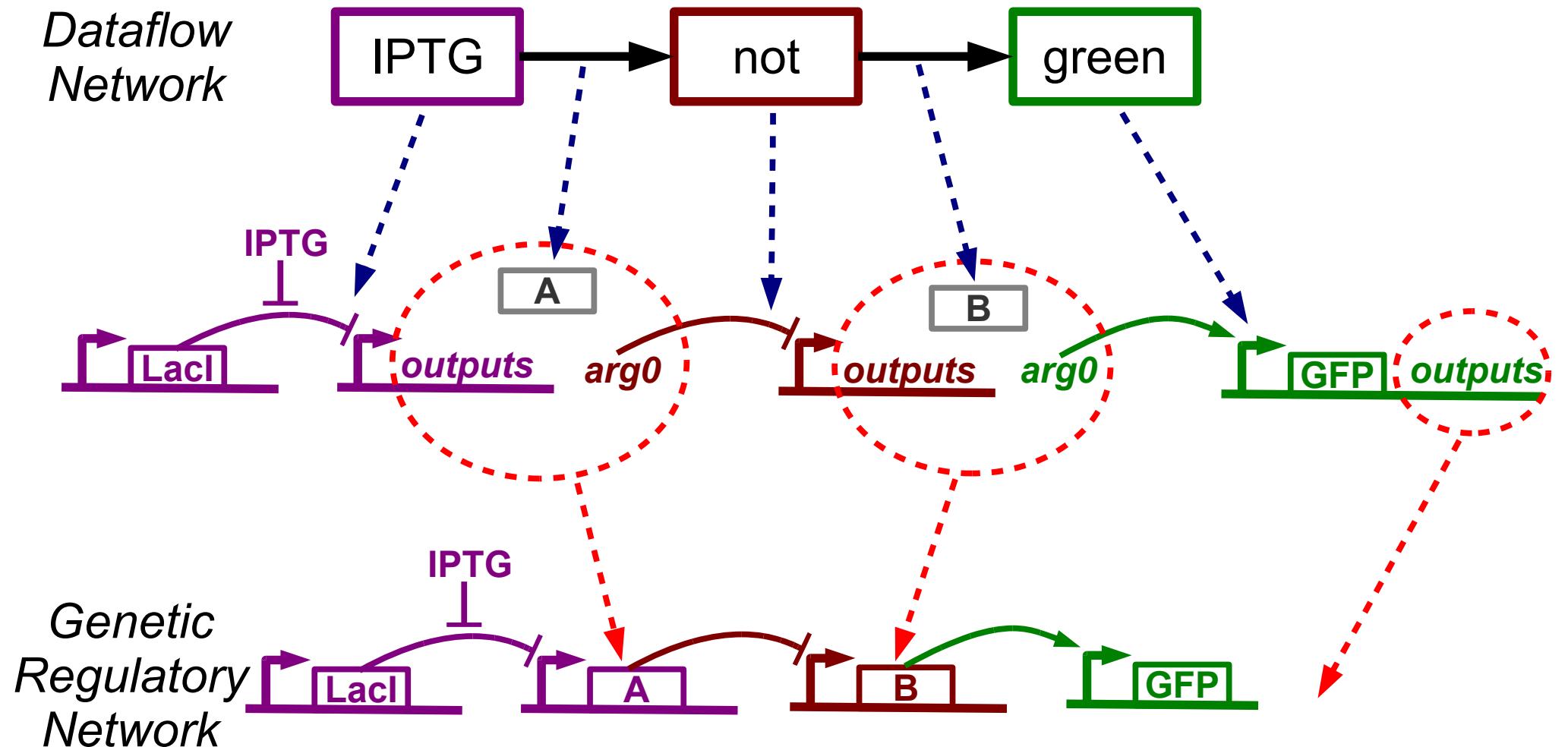


Motif-based Compilation

Dataflow Network

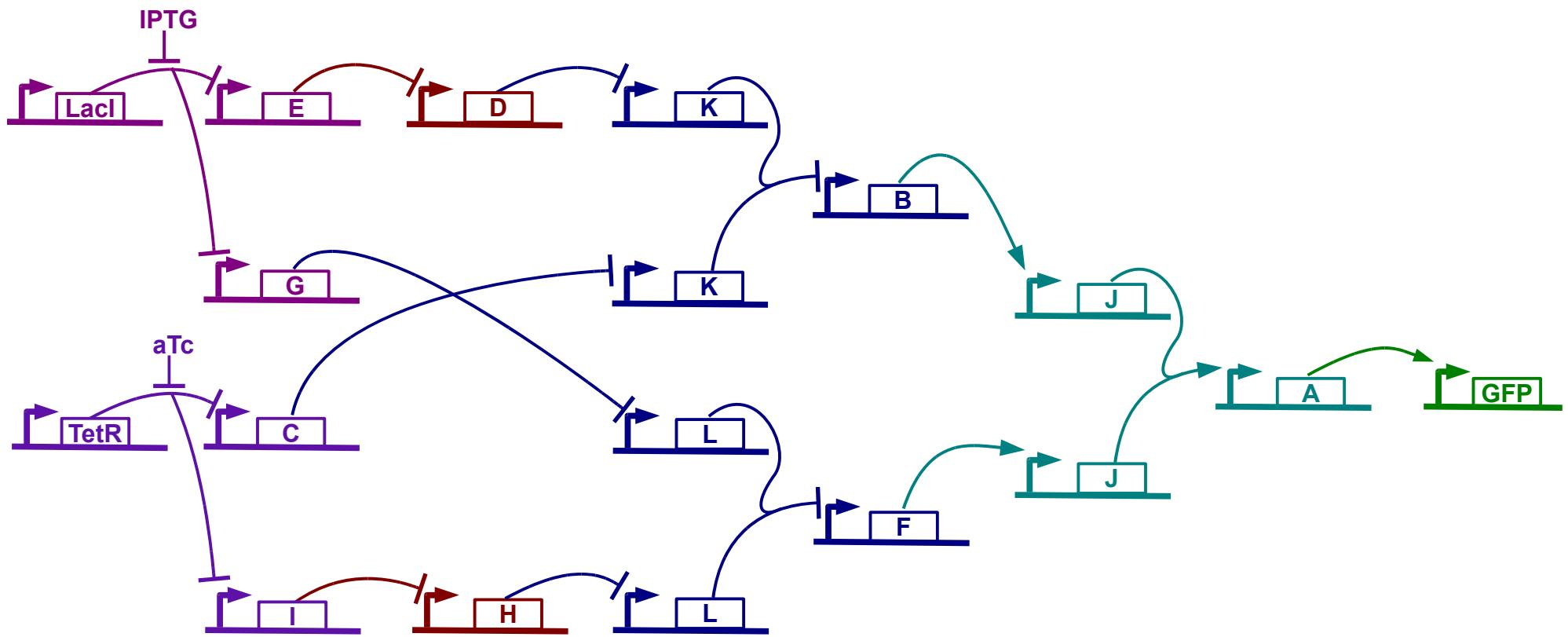


Motif-based Compilation



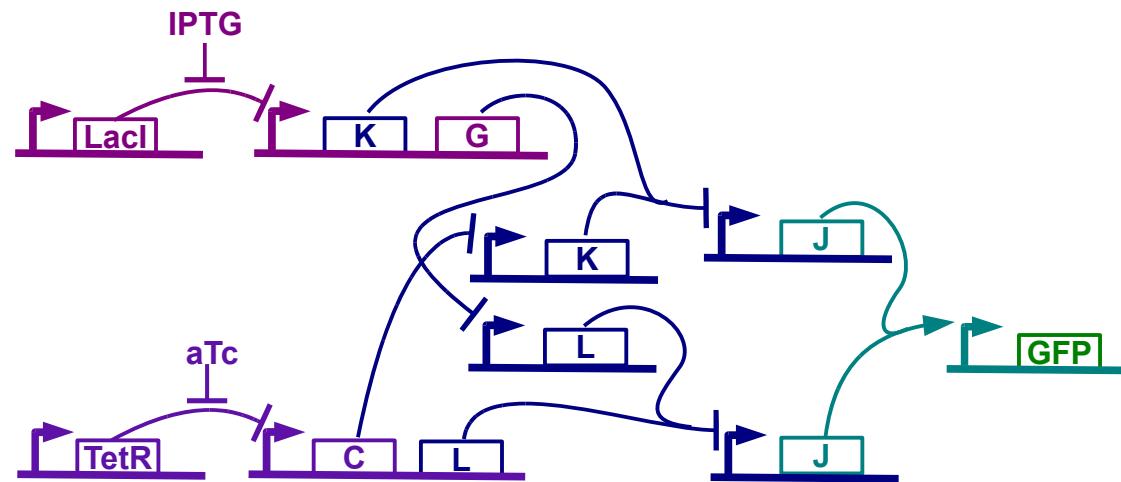
Classical Optimization can be Adapted

- Example: XOR circuit



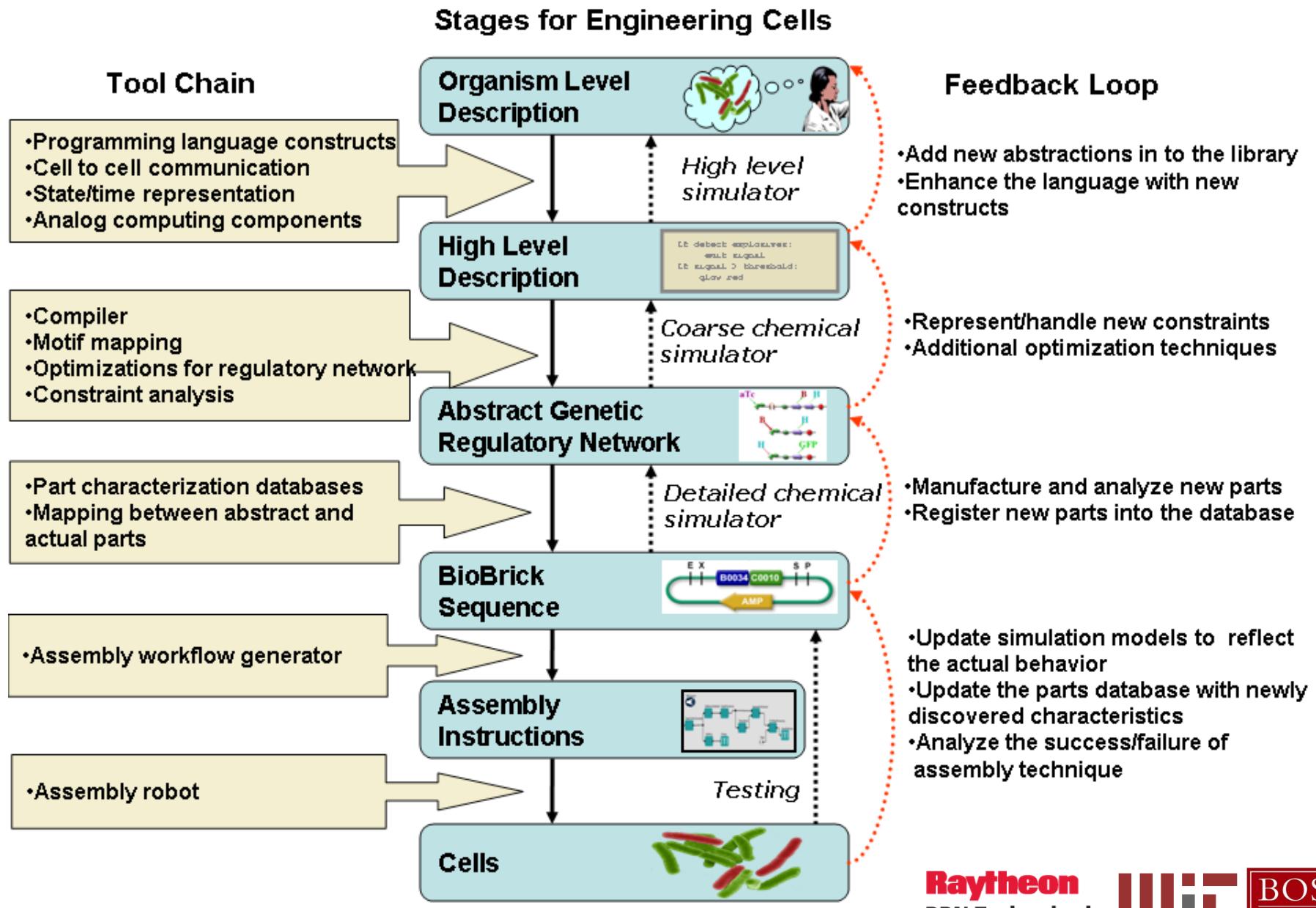
Classical Optimization can be Adapted

- Example: XOR circuit

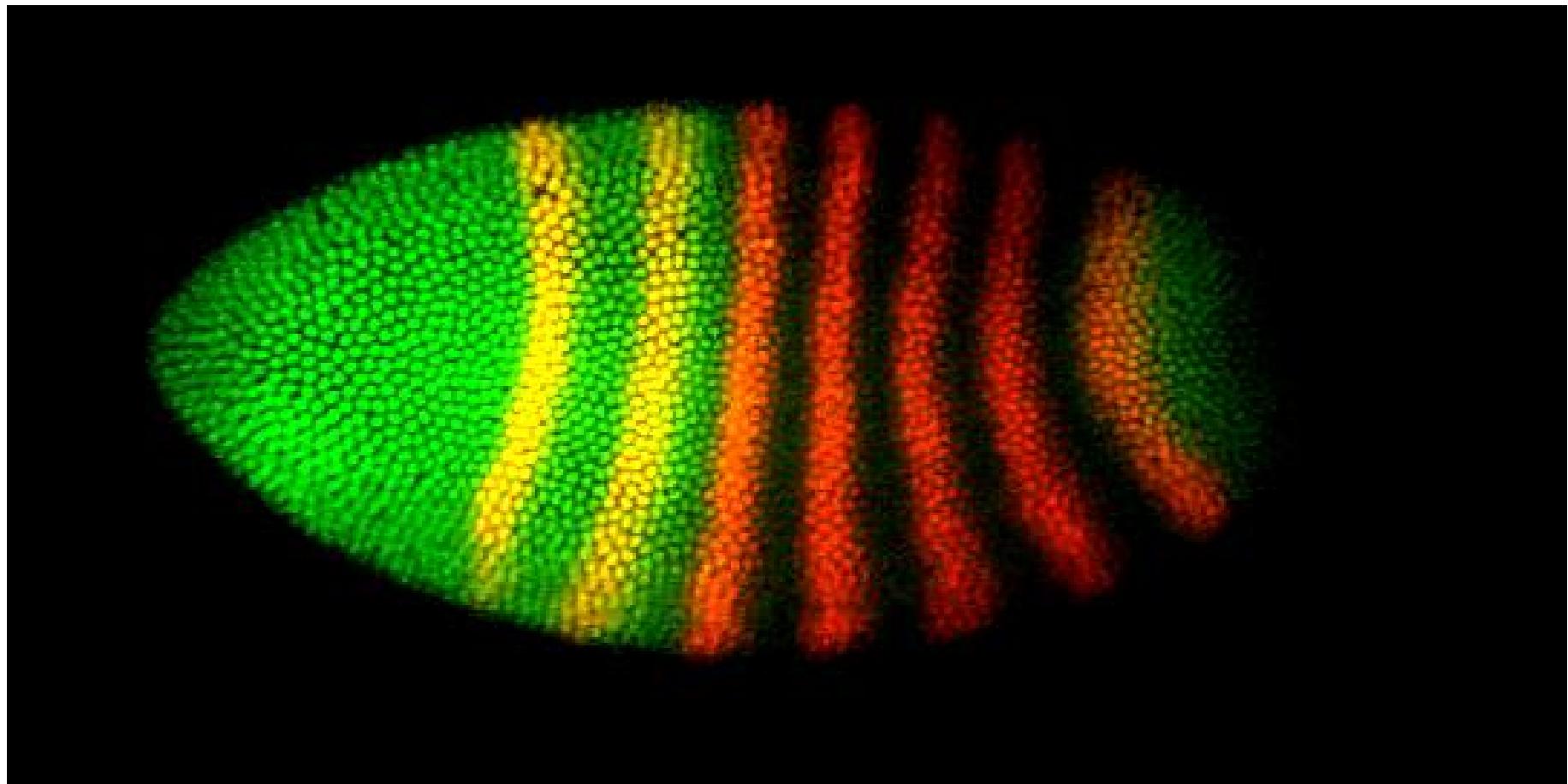


After optimization: ~50% improvement

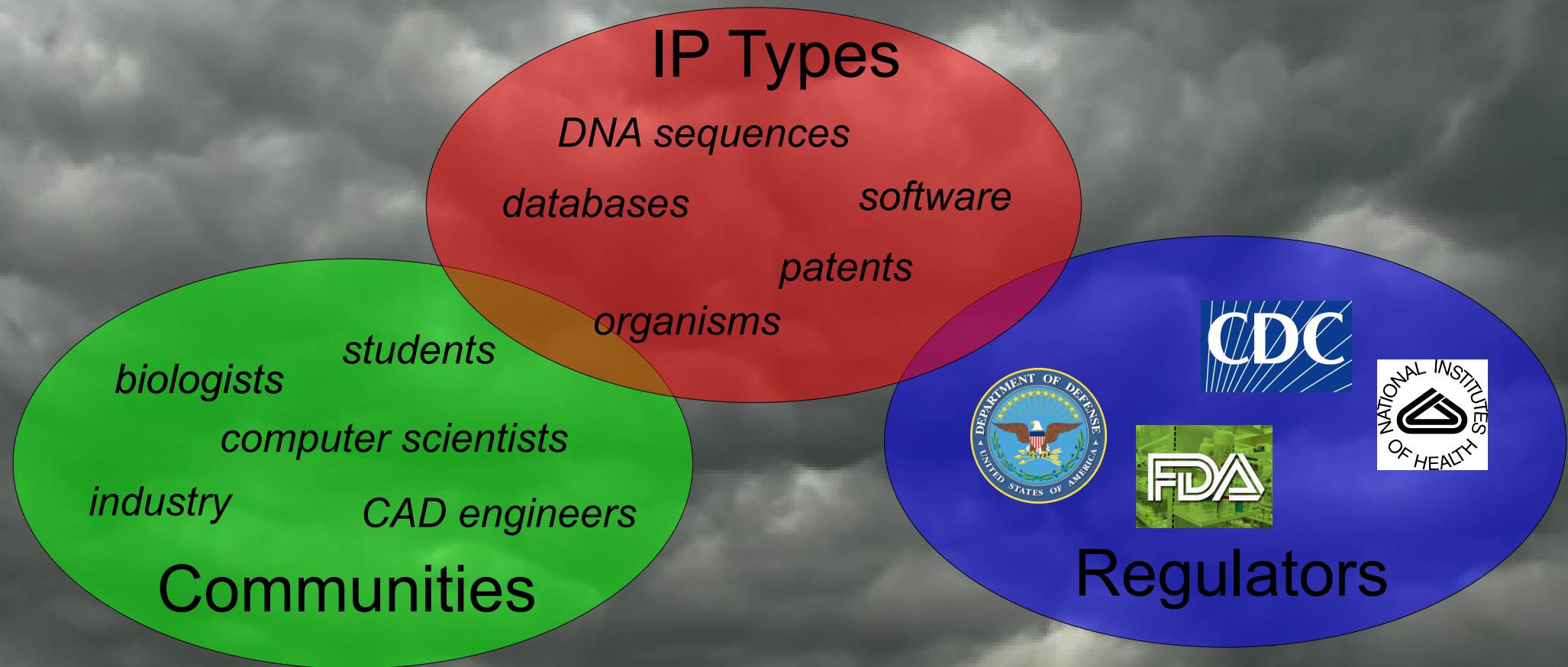
End-to-End Toolchain Project



And on to larger organisms...



IP Challenges



Many components; integration with vendors

Full automation: no human interpretation

Summary

- Proto allows complex spatial computing problems to be solved with simple programs.
- Proto & other approaches beginning to link together to automate synthetic biology
- Major IP thunderclouds on the horizon...

Proto is available

<http://proto.bbn.com>

(or google “MIT Proto”)

- Includes libraries, compiler, kernel, simulator, platforms, tutorial
- Licensed under GPL (w. libc-type exception)