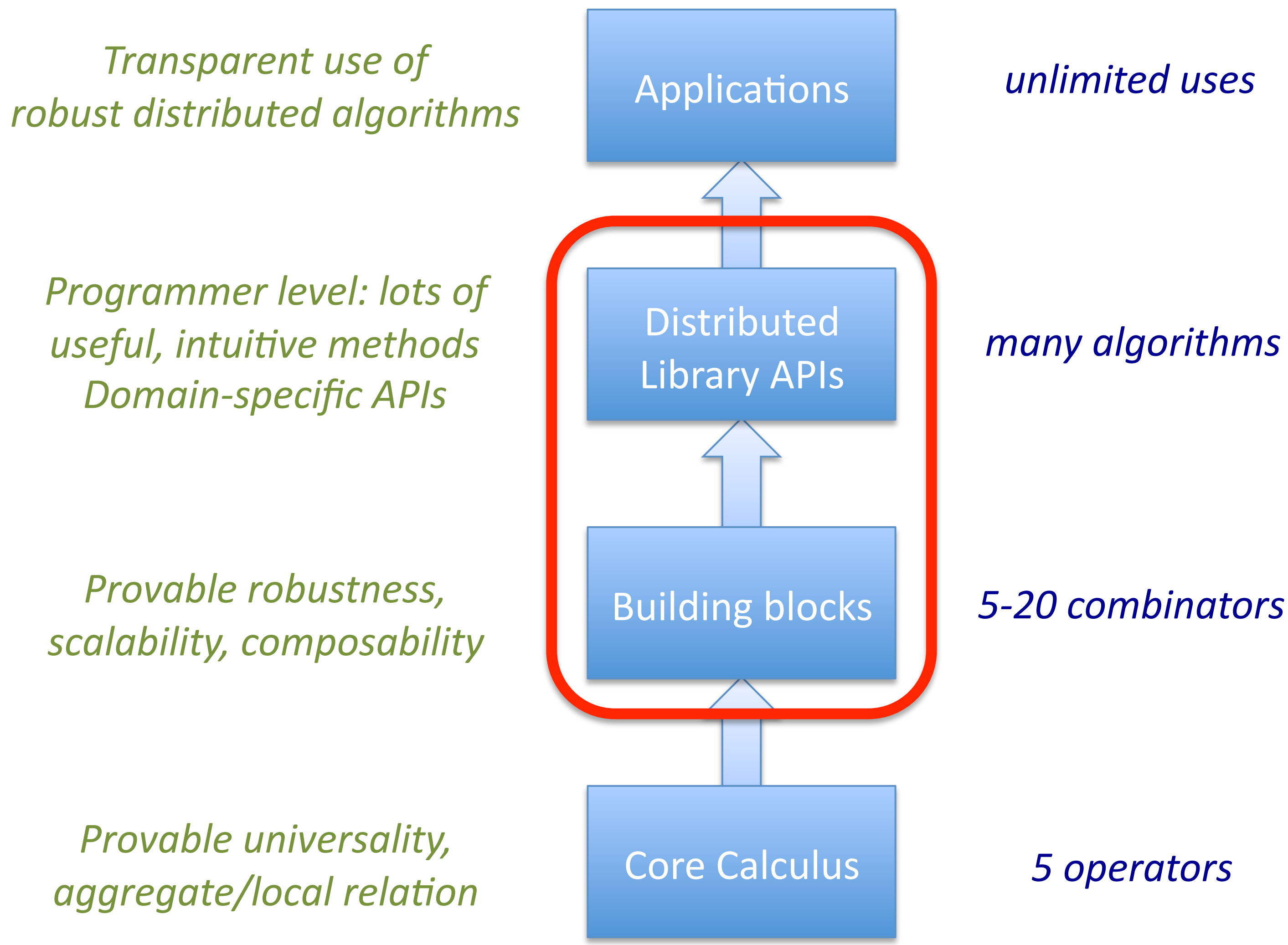
From Building Block Algebras to Adaptive Libraries

Jacob Beal, Mirko Viroli

Tutorial at 8th IEEE SASO
Imperial College London
September 2014

Raytheon
BBN Technologies

From Principle to Practice:



Outline

- **Building Blocks**
- Space-Time Universality
- Eventual Consistency

Example: Managing Crowd Danger

Foundation: Field Calculus

Raytheon BBN Technologies

Variable　Literal Value　Local Ops　State

Communication

$$
\begin{array}{llr}
\mathtt{e} & ::= \mathtt{x} \mid \mathtt{l} \mid (\mathtt{b}\ \overline{\mathtt{e}}) \mid (\mathtt{f}\ \overline{\mathtt{e}}) & \text{expression} \\
 & \mid (\mathtt{rep}\ \mathtt{x}\ \mathtt{w}\ \mathtt{e}) \mid (\mathtt{nbr}\ \mathtt{e}) \mid (\mathtt{if}\ \mathtt{e}\ \mathtt{e}\ \mathtt{e}) & \text{special constructs} \\
\mathtt{w} & ::= \mathtt{x} \mid \mathtt{l} & \text{variable or value} \\
\mathtt{F} & ::= (\mathtt{def}\ \mathtt{f}(\overline{\mathtt{x}})\ \mathtt{e}) & \text{function} \\
\mathtt{P} & ::= \overline{\mathtt{F}}\ \mathtt{e} & \text{program}
\end{array}
$$

Domain Change

Abstract & Call Functions

- Aggregate/local coherence, space-time universal
- Too low level, no adaptivity guarantees

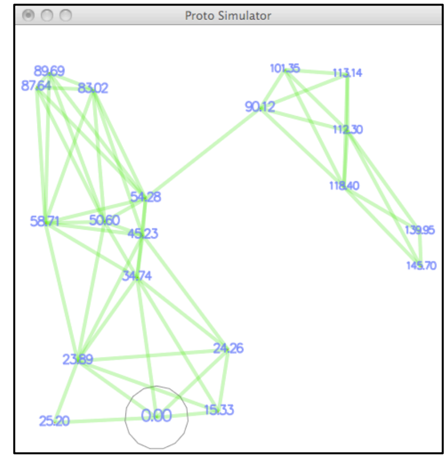
[Viroli et al., ’13]

Making a library of building blocks:

Raytheon BBN Technologies

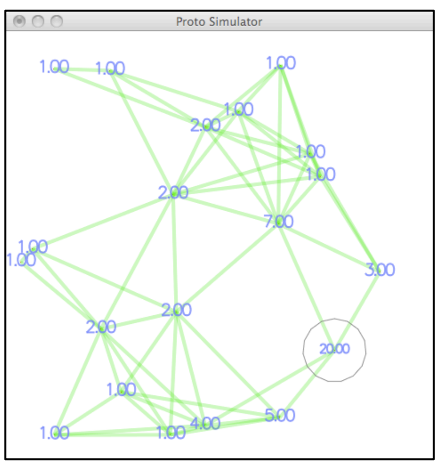
Function	Space	Time
Structure	`nbr-range,...`	`dt,...`
Aggregation	`C`	`T`
Spreading	`G`	
Symmetry breaking	`S`	random
Restriction	`if`	
Compute	computable functions, random	

All compositions are self-stabilizing

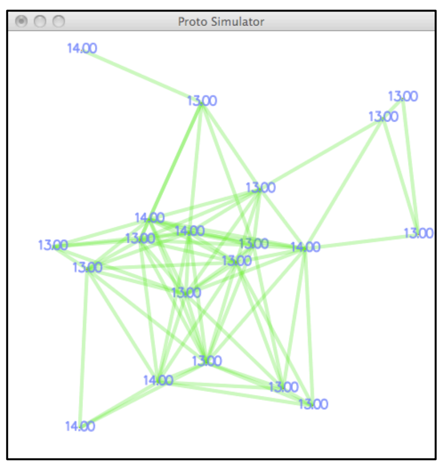


Time-decay

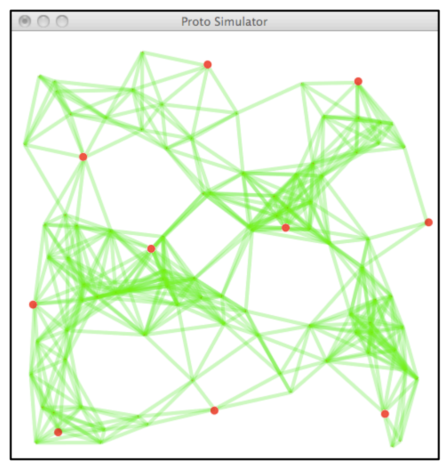
Converge-cast



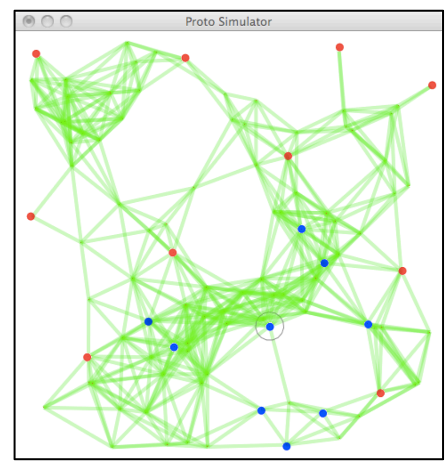
Gradient-path-integral



Sparse-choice



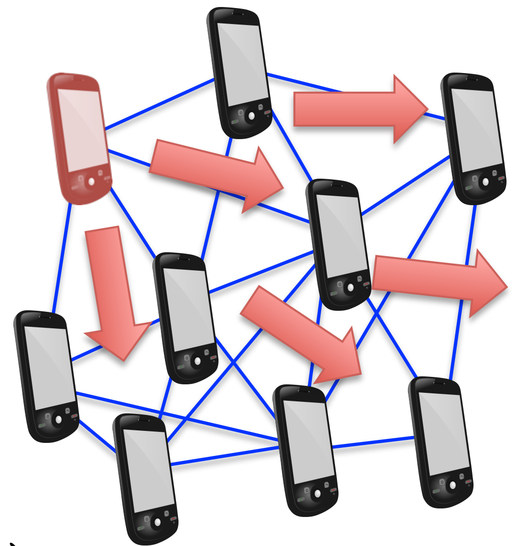
`if`



Building Block: G

Raytheon
BBN Technologies

Information spreading



Field Calculus Implementation:

```
(def G (source initial metric accumulate)
 (2nd
  (rep distance-value
   (tuple infinity initial)
   (mux source (tuple 0 initial)
    (min-hood
      (tuple
       (+ (1st (nbr distance-value)) (metric))
       (accumulate (2nd (nbr distance-value)))))))))
```

Library Examples:

```
(def distance-to (source)
 (G source 0 nbr-range (fun (v) (+ v (nbr-range)))))
```

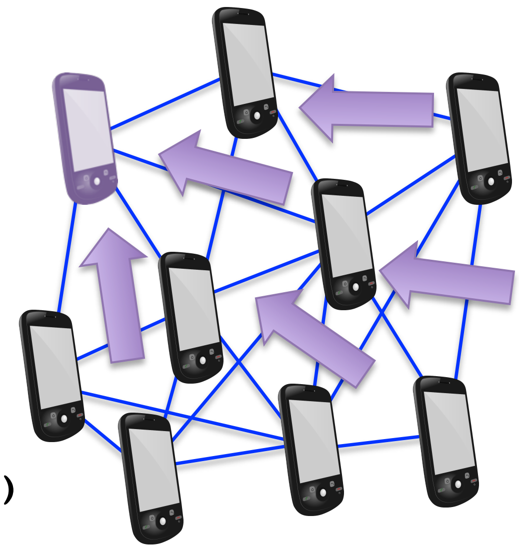
```
(def broadcast (source value)
 (G source value nbr-range identity))
```

Building Block: C

Information collection

Field Calculus Implementation:

```
(def C (potential accumulate local null)
 (rep v local
  (accumulate local
   (accumulate-hood accumulate
    (mux (= (nbr (find-parent potential)) (uid))
     (nbr v) null)))))
(def find-parent (potential)
 (mux (< (1st (min-hood (nbr potential))) potential)
   (2nd (min-hood (nbr (tuple potential (uid)))))
   NaN))
```



Library Examples:

```
(def summarize (sink accumulate local null)
 (broadcast sink
  (C (distance-to sink) accumulate local null)))

(def average (sink value)
  (/ (summarize sink + value 0)
     (summarize sink + 1 0)))
```

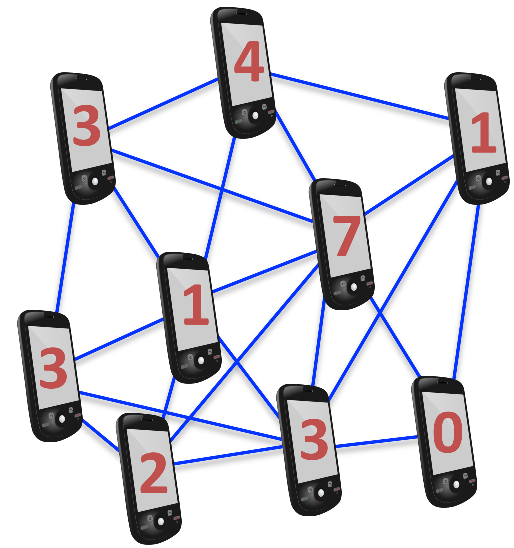
Building Block: T

Raytheon
BBN Technologies

Time-summarization of information

Field Calculus Implementation:

```
(def T (initial decay)
 (rep v initial
  (min initial
   (max 0 (decay v)))))
```



Library Examples:

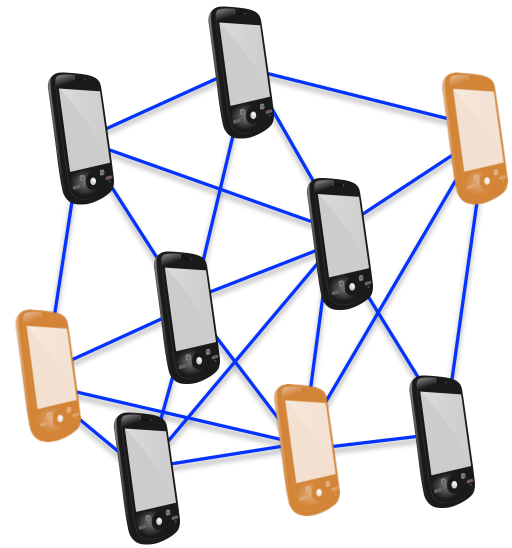
```
(def timer (length)
 (T length (fun (t) (- t (dt)))))
```

```
(def limited-memory (value timeout)
 (2nd (T (tuple timeout value)
      (fun (t) (tuple (- (1st t) (dt)) (2nd t))))))
```

Building Block: S

Raytheon BBN Technologies

Choice of sparse subset



Field Calculus Implementation:

```
(def S (grain metric)
   (break-using-uids (random-uid) grain metric))

(def random-uid ()
  (rep v (tuple (rnd 0 1) (uid))
   (tuple (1st v) (uid))))

(def break-using-uids (uid grain metric)
 (= uid
   (rep lead uid
    (distance-competition
     (G (= uid lead) 0 metric
      (fun (v) (+ v (metric))))
     lead uid grain metric))))

(def distance-competition (d lead uid grain metric)
  (mux (> d grain) uid
   (mux (>= d (* 0.5 grain)) infinity
    (min-hood
     (mux (>= (+ (nbr d) (metric)) (* 0.5 grain))
      infinity
      (nbr lead))))))
```

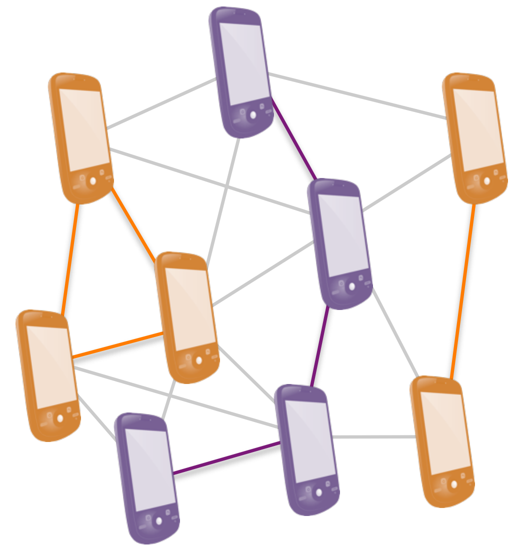
Building Block: `if`

Raytheon
BBN Technologies

Restrict scope to subspaces

Field Calculus Implementation:

```
(if test
    true-expression
    false-expression)
```



Library Examples:

```
(def distance-avoiding-obstacles (source obstacles)
  (if obstacles
      infinity
      (distance-to source)))


(def recent-event (event timeout)
  (if event true (> (timer timeout) 0)))
```

Applying building blocks:

Raytheon
BBN Technologies

Example API algorithms from building blocks:

distance-to (source)	max-likelihood (source p)
broadcast (source value)	path-forecast (source obstacle)
summarize (sink accumulate local null)	average (sink value)
integral (sink value)	region-max (sink value)
timer (length)	limited-memory (value timeout)
random-voronoi (grain metric)	group-size (region)
broadcast-region (region source value)	recent-event (event timeout)
distance-avoiding-obstacles (source obstacles)	

Since based on these 5 building blocks, all programs built this way are self-stabilizing!

Complex Example: Crowd Management

Raytheon BBN Technologies

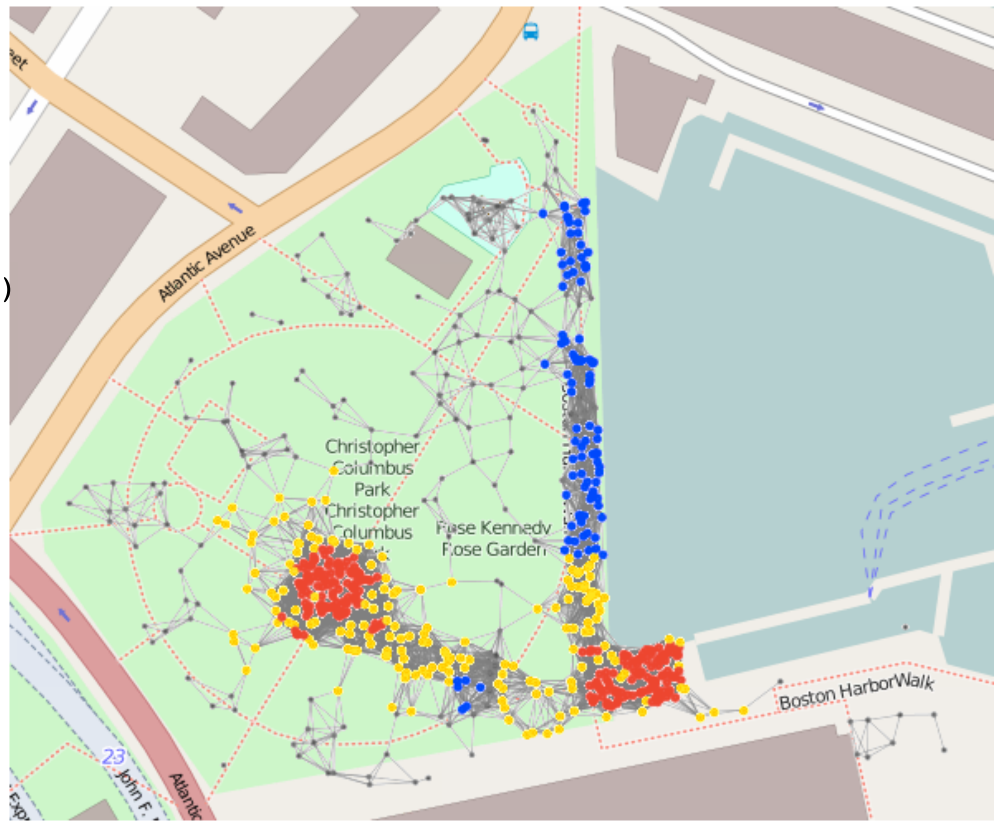
```
(def crowd-tracking (p)
 ;; Consider only Fruin LoS E or F within last minute
 (if (recently-true (> (density-est p) 1.08) 60)
    ;; Use S to break into "cells" and estimate danger of each
    (+ 1 (dangerous-density (S 30) p))
    0))

(def recently-true (state memory-time)
 ;; Make sure first state is false, not true...
 (rt-sub (not (T 1 1)) state memory-time))
(def rt-sub (started s m)
 (if state 1 (limited-memory s m)))

(def dangerous-density (partition p)
 ;; Only dangerous if above critical density threshold...
 (and
  (> (average partition (density-est p)) 2.17)
  ;; ... and also involving many people.
  (> (summarize partition + (/ 1 p) 0) 300)))
```

```
(def crowd-warning (p range)
  (> (distance-to (= (crowd-tracking p) 2))
   range)

(def safe-navigation (destination p)
  (distance-avoiding-obstacles
   destination (crowd-warning p)))
```



18 lines non-whitespace code

10 library calls (18 ops)

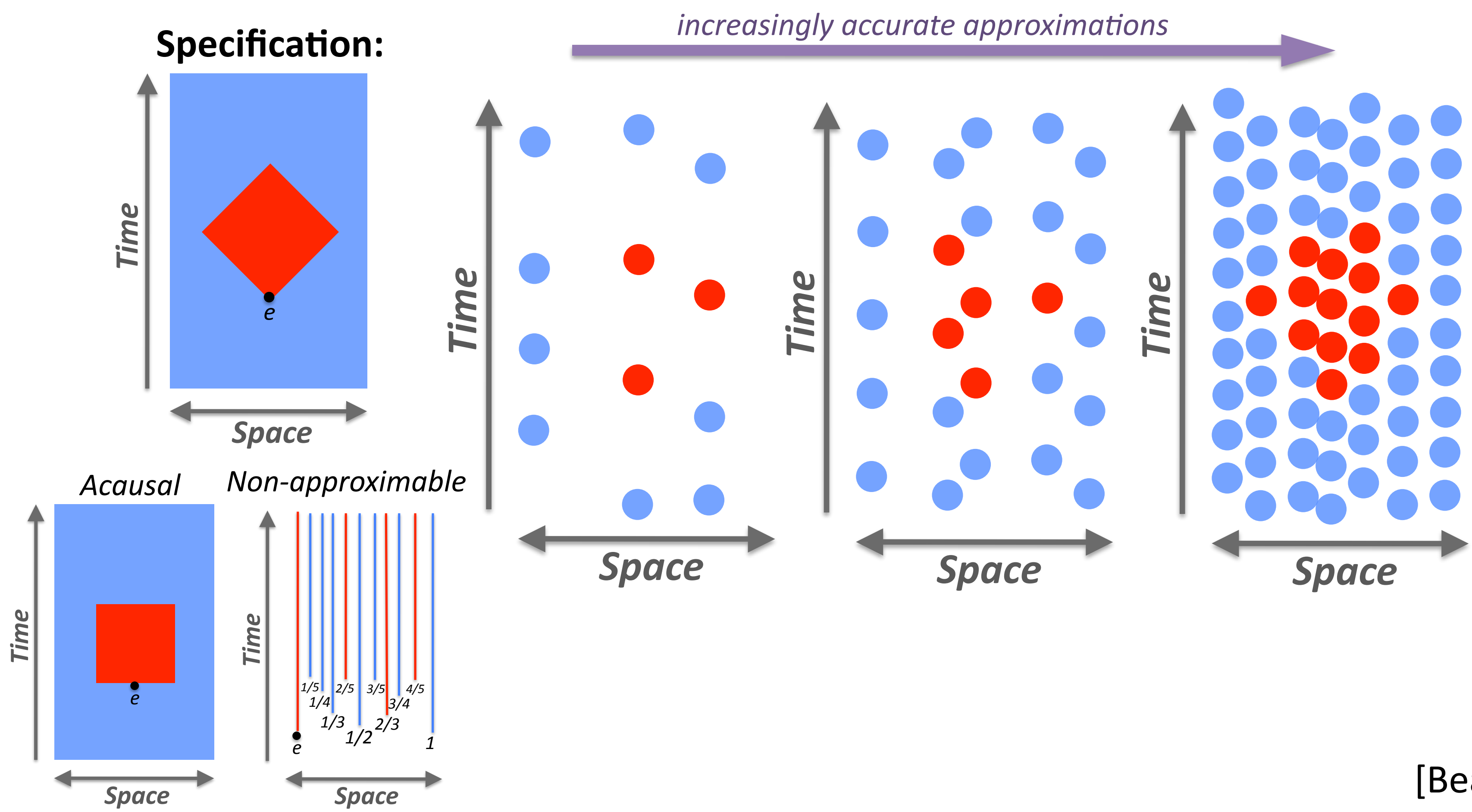
if: 3 G: 9 C: 3 T: 2 S: 1

Outline

- Building Blocks
- **Space-Time Universality**
- Eventual Consistency

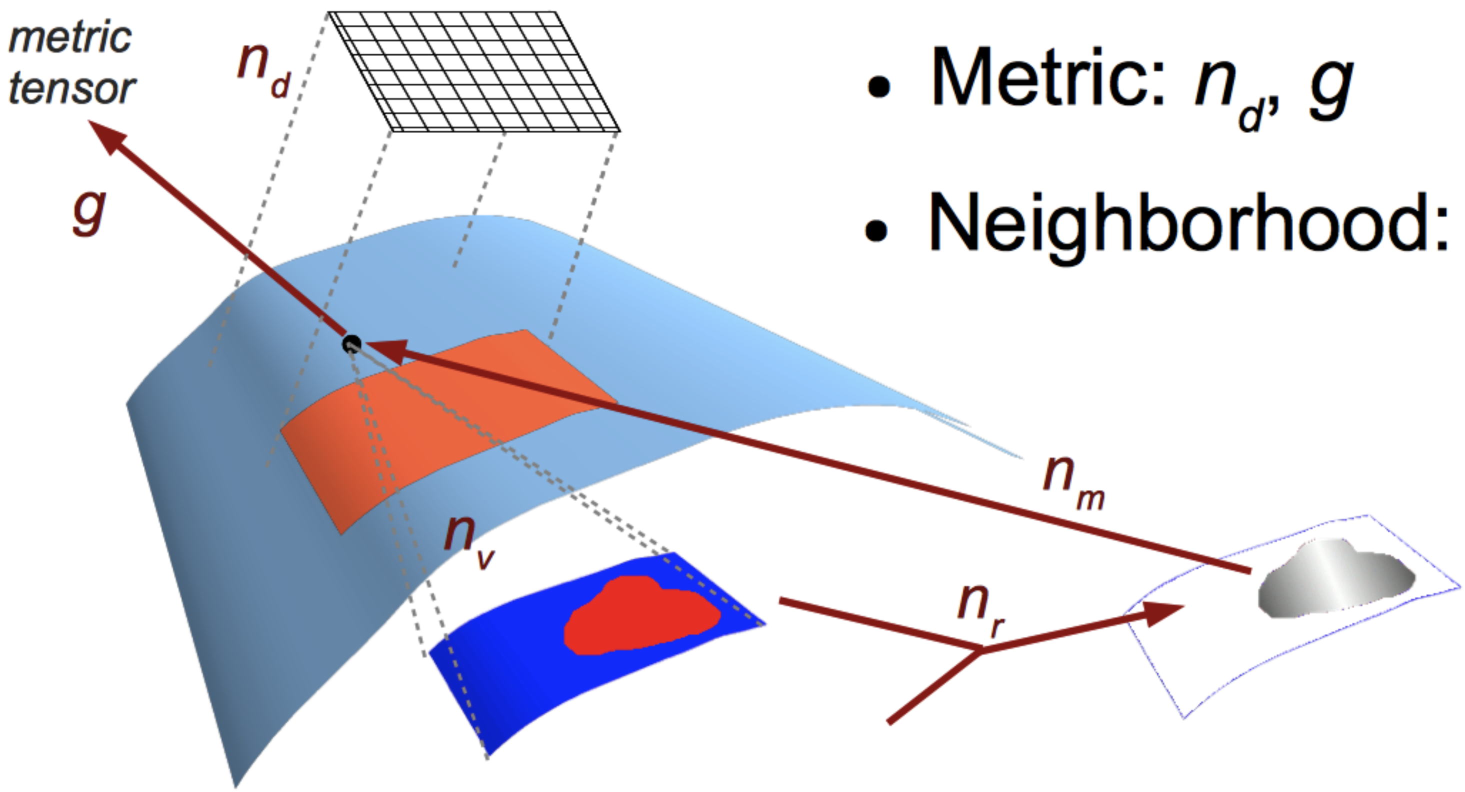
Space-Time Universality

Space-time Universal = arbitrarily good approximation of any causal, finitely-approximable computation



[Beal, '10]

Proposed Universal Operators



- Pointwise Turing-universal: P
  - Metric: n_d, g
  - Neighborhood: n_v, n_r, n_m

[Beal, '10]

Proof sketch

Field Calculus

2. Field calculus can approximate those Proto programs

1. Some Proto programs are finitely approximable

Proto

3. This subset is enough to get universality

Space-Time Universality

Field Calculus

Raytheon
BBN Technologies

Variable   Literal Value   Local Ops

State

Communication

e	::= x | l | (b $\overline{\text{e}}$) | (f $\overline{\text{e}}$)	expression
	| (rep x w e) | (nbr e) | (if e e e)	special constructs
w	::= x | l	variable or value
F	::= (def f($\overline{\text{x}}$) e)	function
P	::= $\overline{\text{F}}$ e	program

Domain Change

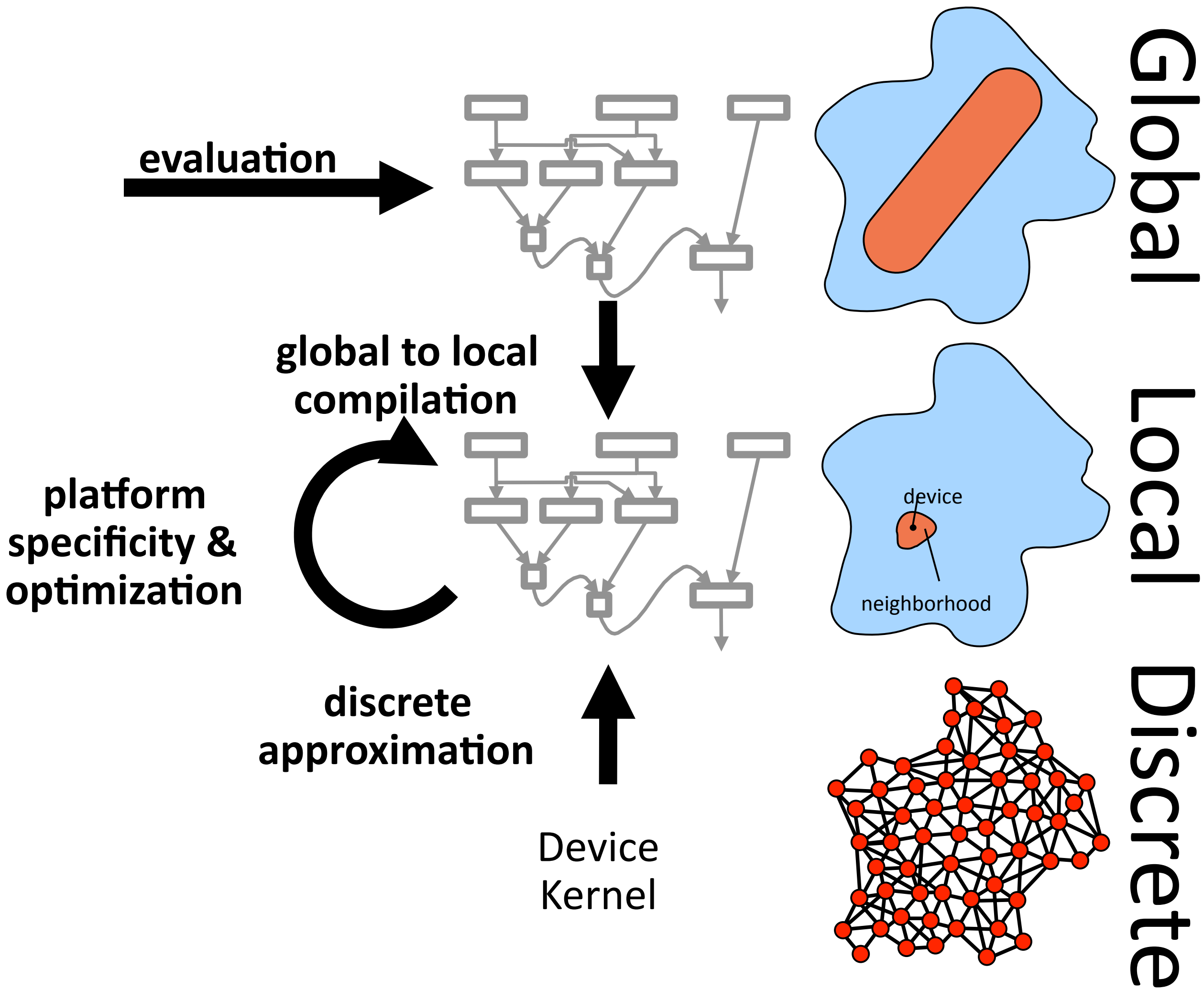
Abstract & Call Functions

[Viroli et al., ’13]

Proto

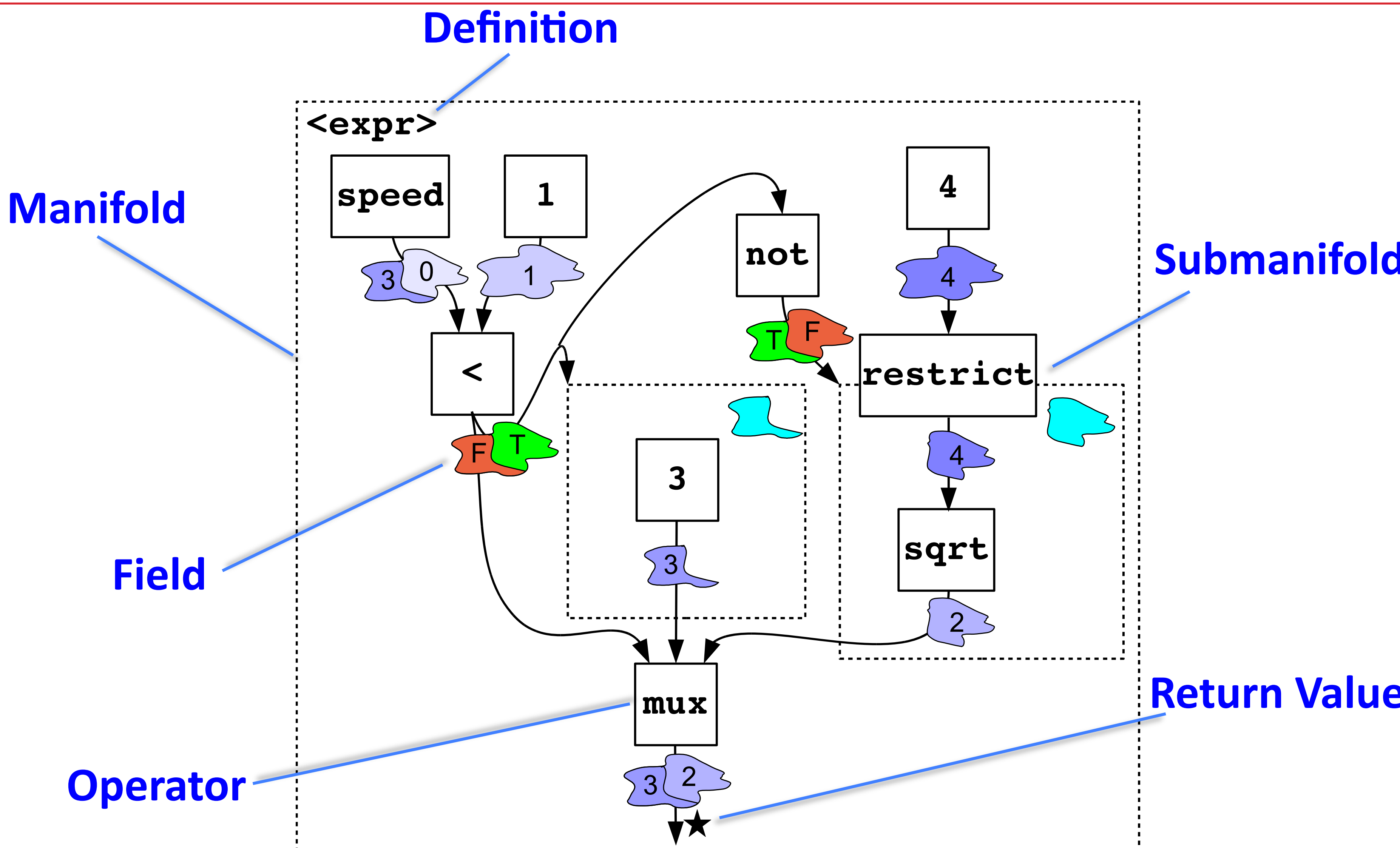
Raytheon
BBN Technologies

```
(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]

Proto as Dataflow Graph



Well-defined program = no inconsistencies in domains, graph structure

Proof sketch

Raytheon
BBN Technologies

Field Calculus

2. Field calculus can approximate those Proto programs

1. Some Proto programs are finitely approximable

Proto

3. This subset is enough to get universality

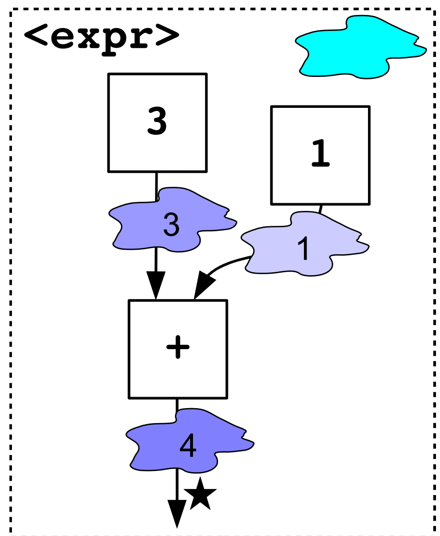
Space-Time Universality

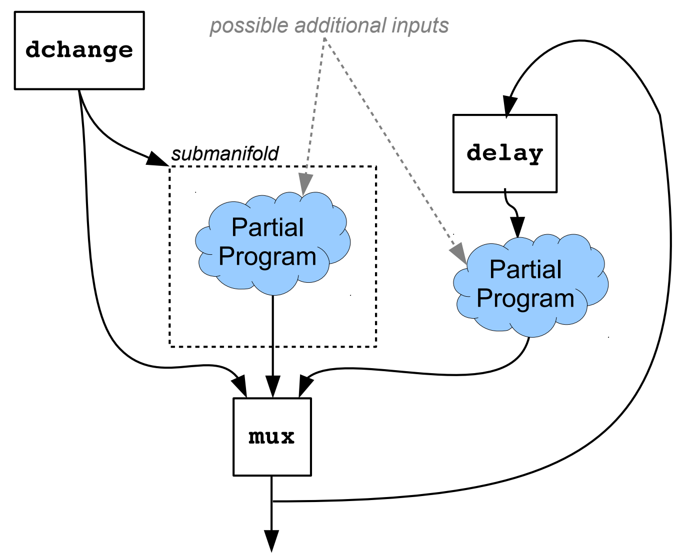
Theorem 1: Proto approximability

Raytheon BBN Technologies

Any well-defined Proto program P = (M, F, O, R, D) composed only of finitely approximable operator instances is finitely approximable.

If inputs converge, then output converges





Intuition: feed-forward composition + special forms

Note: some surprising things (e.g., '=') aren't approximable!

Neighborhood Independence

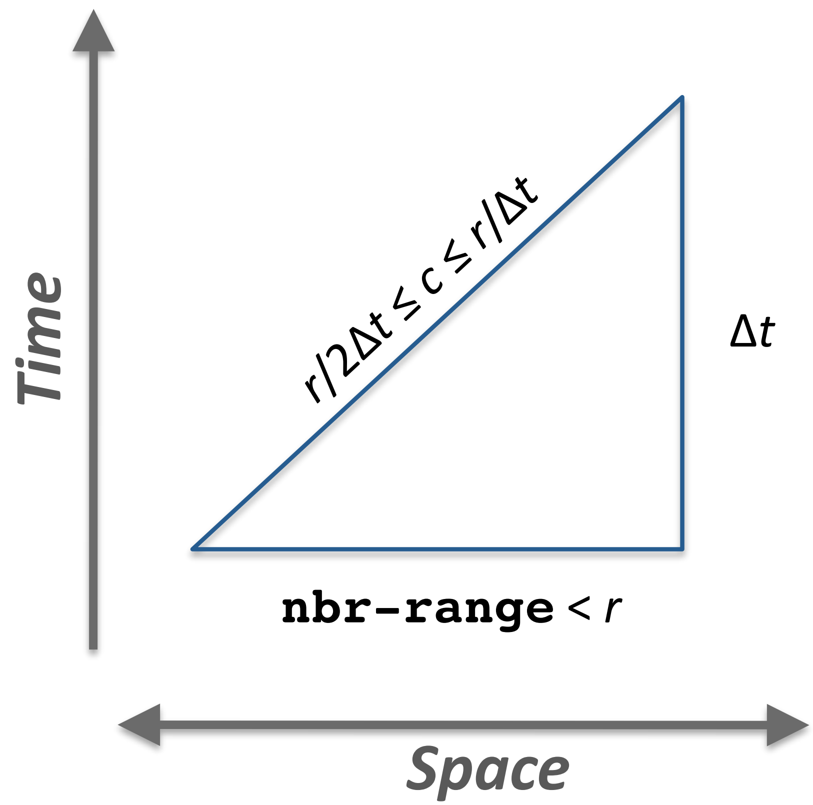
Raytheon
BBN Technologies

- For consistent communication speed, neighborhood must shrink with time-step.

A problem program:

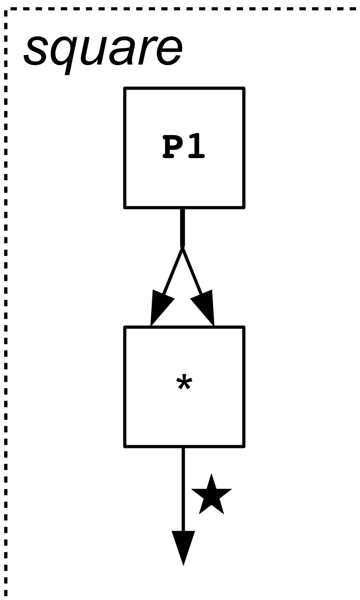
```
(def use-speed-and-radius (bool)
  (* (any-hood (nbr bool))
     (distance-to bool)))
```

- Neighborhood-independent programs aren't affected by this problem



Theorem 2: Field Calculus → Proto

Any well-defined neighborhood-independent Proto program P = (M, F, O, R, D) composed only of finitely approximable operator instances can be approximated using field calculus.




```
(def function_1 (p_1)
  (function_1_1 p_1))

(def function_1_1 (p_1)
  (function_1_2 p_1 (* p_1 p_1)))

(def function_1_2 (p_1 v_1) v_1)
```

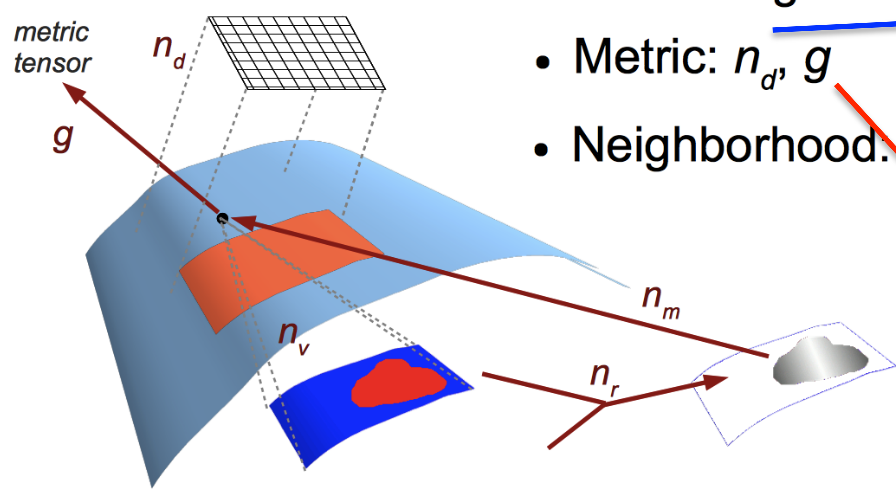
Intuition: construct equivalent graph in field calculus

Theorem 3: Field Calculus Universality

Raytheon BBN Technologies

Field Calculus is Space-Time Universal

Corollary: Any well-defined finitely approximable Proto program can be approximated by field calculus.



- Pointwise Turing-universal: P
  - Metric: n_d, g
  - Neighborhood: n_v, n_r, n_m

Approximable subset of Proto covers these

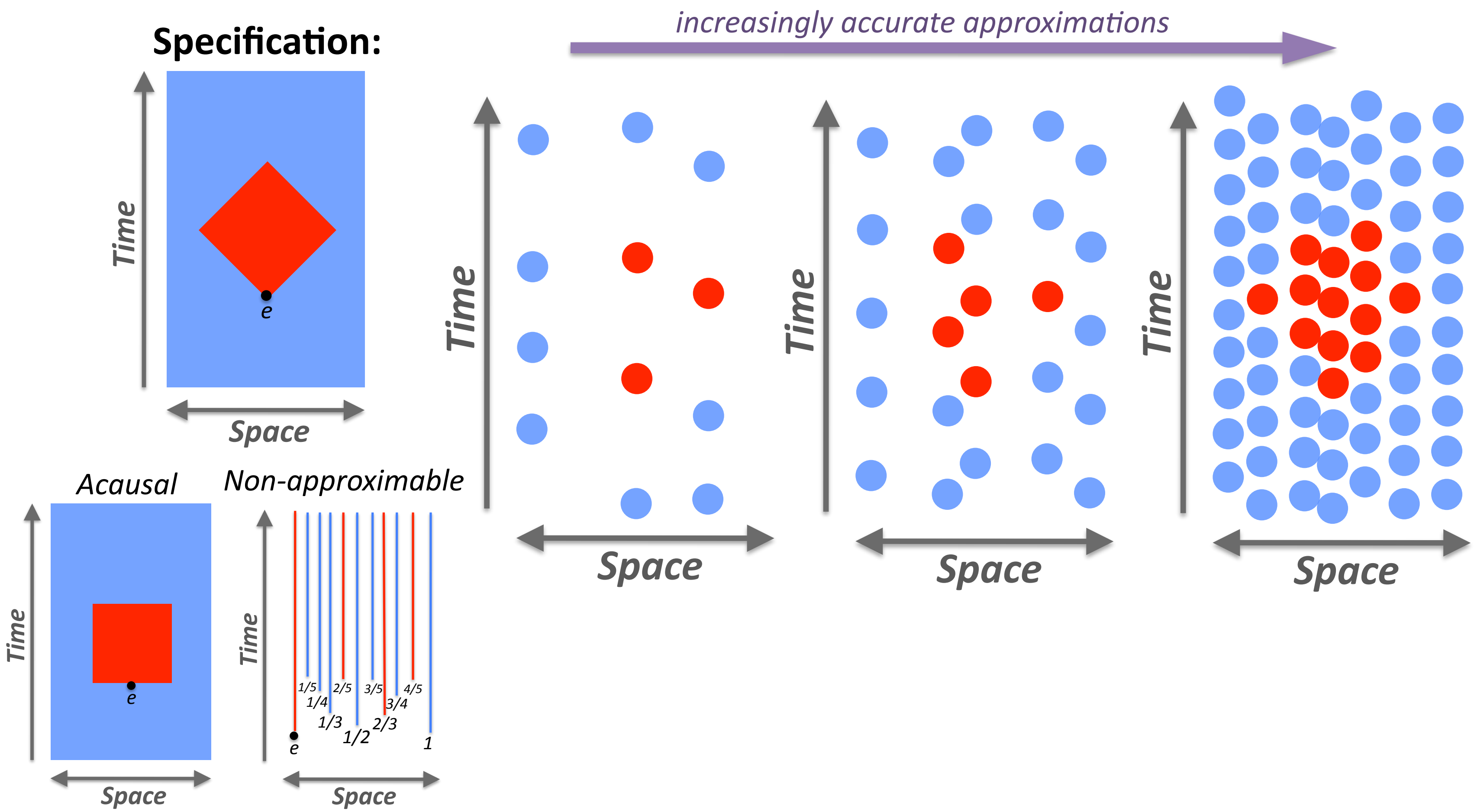
Metric tensor can be a built-in

Intuition: Proto is nearly space-time universal.

Outline

- Building Blocks
- Space-Time Universality
- **Eventual Consistency**

Causality & Finite Approximability



[Beal, ’10]

Eventual Consistency

Raytheon BBN Technologies

- **Consistent Program:** Let P be a space-time program, e be an evaluation environment, and e_i a countable sequence of ε-approximations that approximate field e. Program P is consistent if $P(e_i)$ approximates $P(e_i)$ for every e and e_i.

- **Eventually Consistent Program:** Consider a causal program P evaluated on environment e with domain M. Program P is eventually consistent if, for any environment e in which there is a spatial section S_M such that the values of e do not change at any device in the time-like future $T^+(S_M)$, there is always some spatial section S_M' such that P is consistent on the time-like future $T^+(S_M')$

Intuition: resilience against scale and discretization

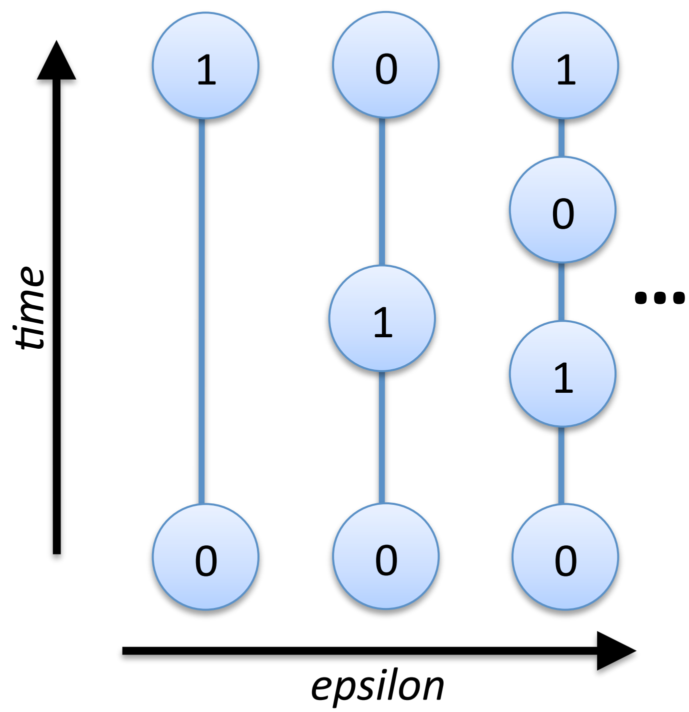
[Beal et al, submitted]

What are the threats to consistency?

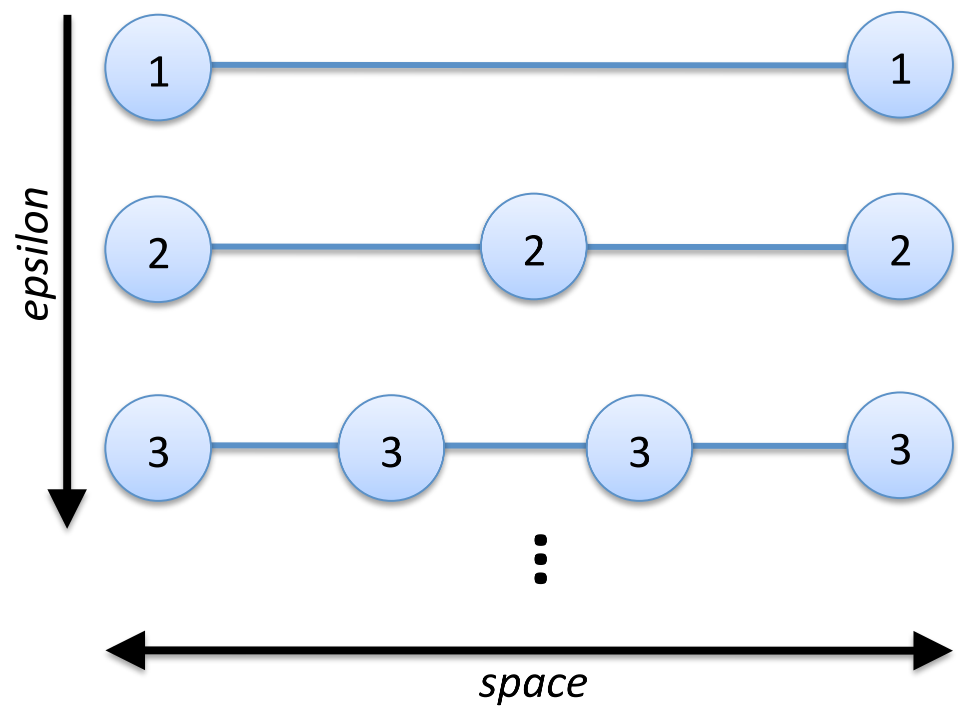
Raytheon BBN Technologies

- Unbounded recursion
- Direct use of **rep**, **nbr** constructs

`(rep x 0 (- 1 x))`



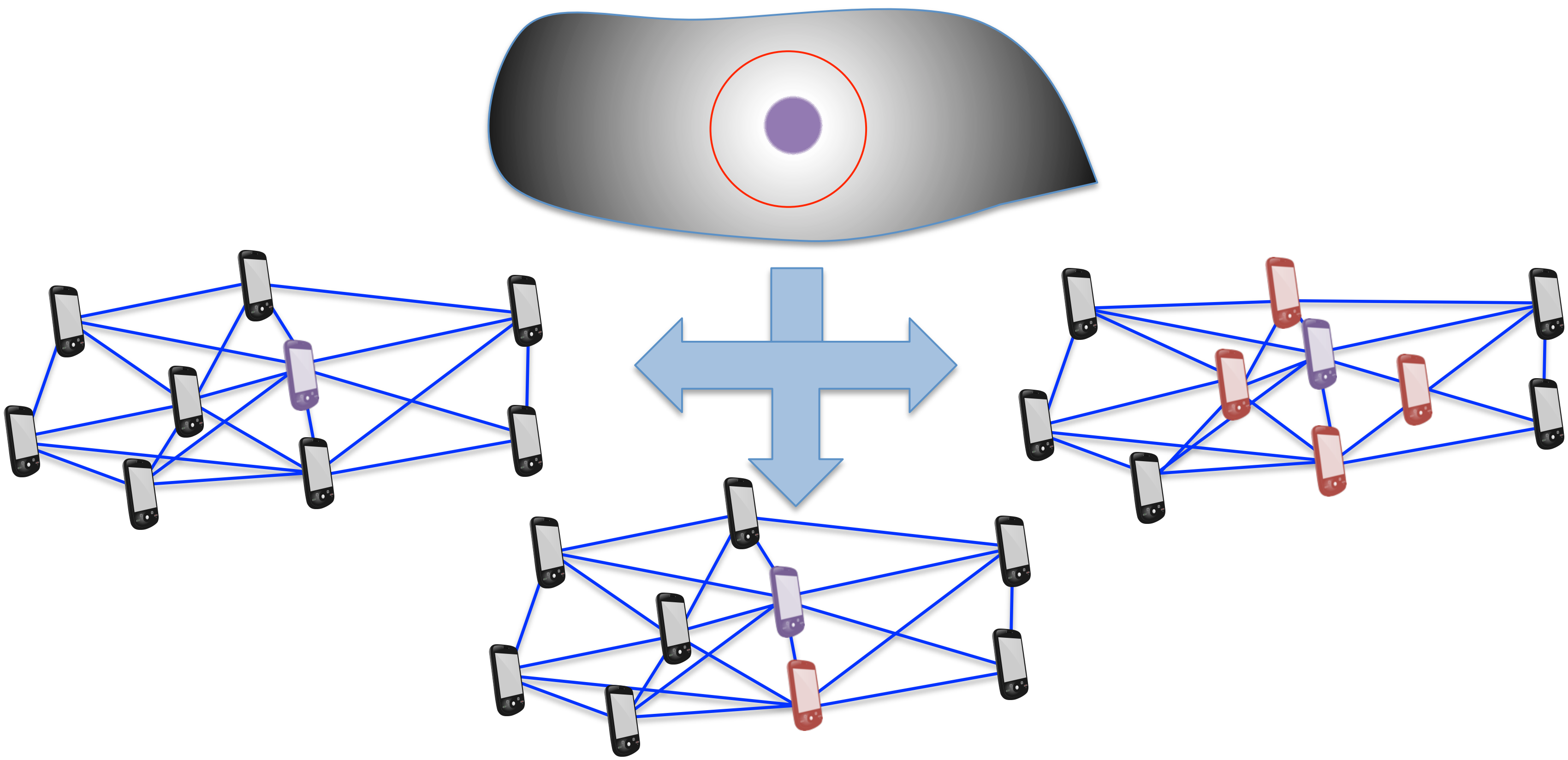
`(/ 1 (min-hood (nbr-range)))`



What are the threats to consistency?

- Fragile values (measure zero sets)

```
(def ring (src)
  (= (distance-to src) 15.0)))
```



GPI-calculus

Raytheon
BBN Technologies

Special "Boundary" value

Only integers and reals

Restricted built-in ops

$$
\begin{array}{lll}
\mathtt{l} ::= \mathbb{B} \mid \mathbb{Z} \mid \mathbb{R} & & \text{;; Literals} \\
\mathtt{b} ::= \mathtt{m} \mid \mathtt{mux} \mid \mathtt{<} & & \text{;; local operators} \\
\mathtt{e} ::= \mathtt{x} \mid \mathtt{l} \mid (\mathtt{b}\ \overline{\mathtt{e}}) \mid (\mathtt{f}\ \overline{\mathtt{e}}) \mid (\mathtt{sense}\ \mathbb{Z}^{+}) & & \text{;; expression} \\
\quad \mid (\mathtt{if\ e\ e\ e}) \mid (\mathtt{GPI\ e\ e\ e\ e}) & & \text{;; special constructs} \\
\mathtt{F} ::= (\mathtt{def}\ \mathtt{f}(\overline{\mathtt{x}})\ \mathtt{e}) & & \text{;; function} \\
\mathtt{P} ::= \overline{\mathtt{F}}\ \mathtt{e} & & \text{;; program}
\end{array}
$$

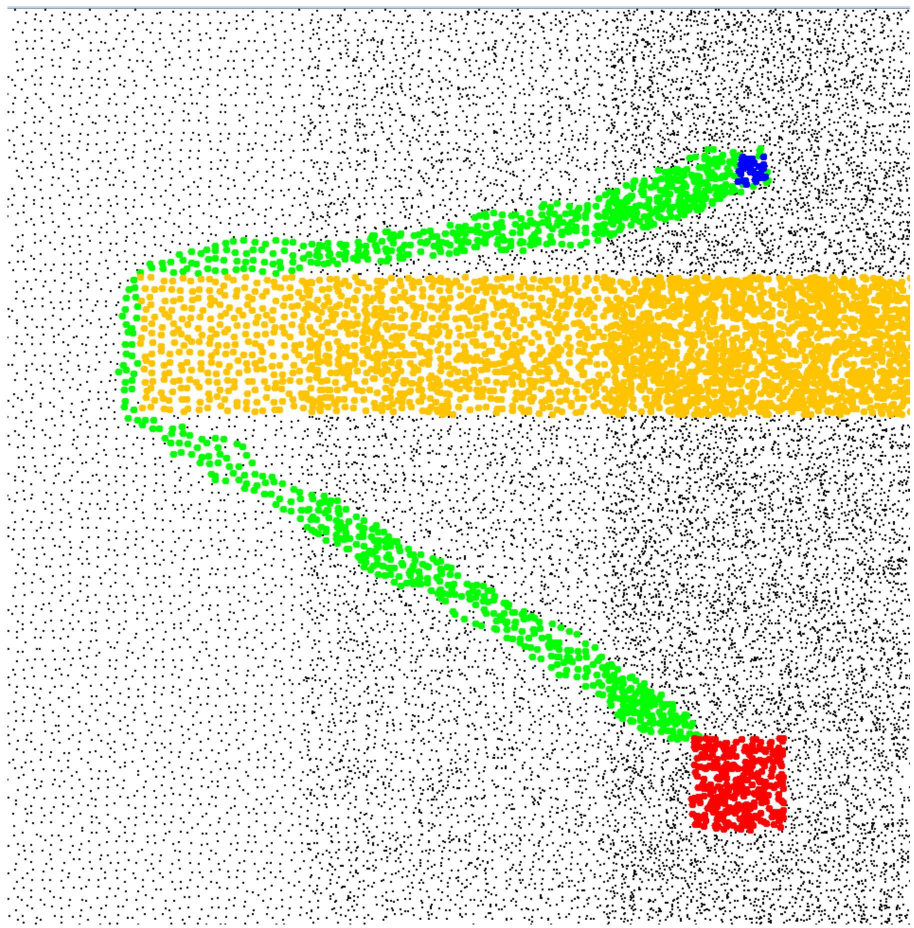
GPI replaces nbr, rep

Semantics prohibits recursion

- Restriction of field calculus to consistent subset
  - Real # comparison produces "Boundary" for equality
  - GPI = Gradient-Path-Integral

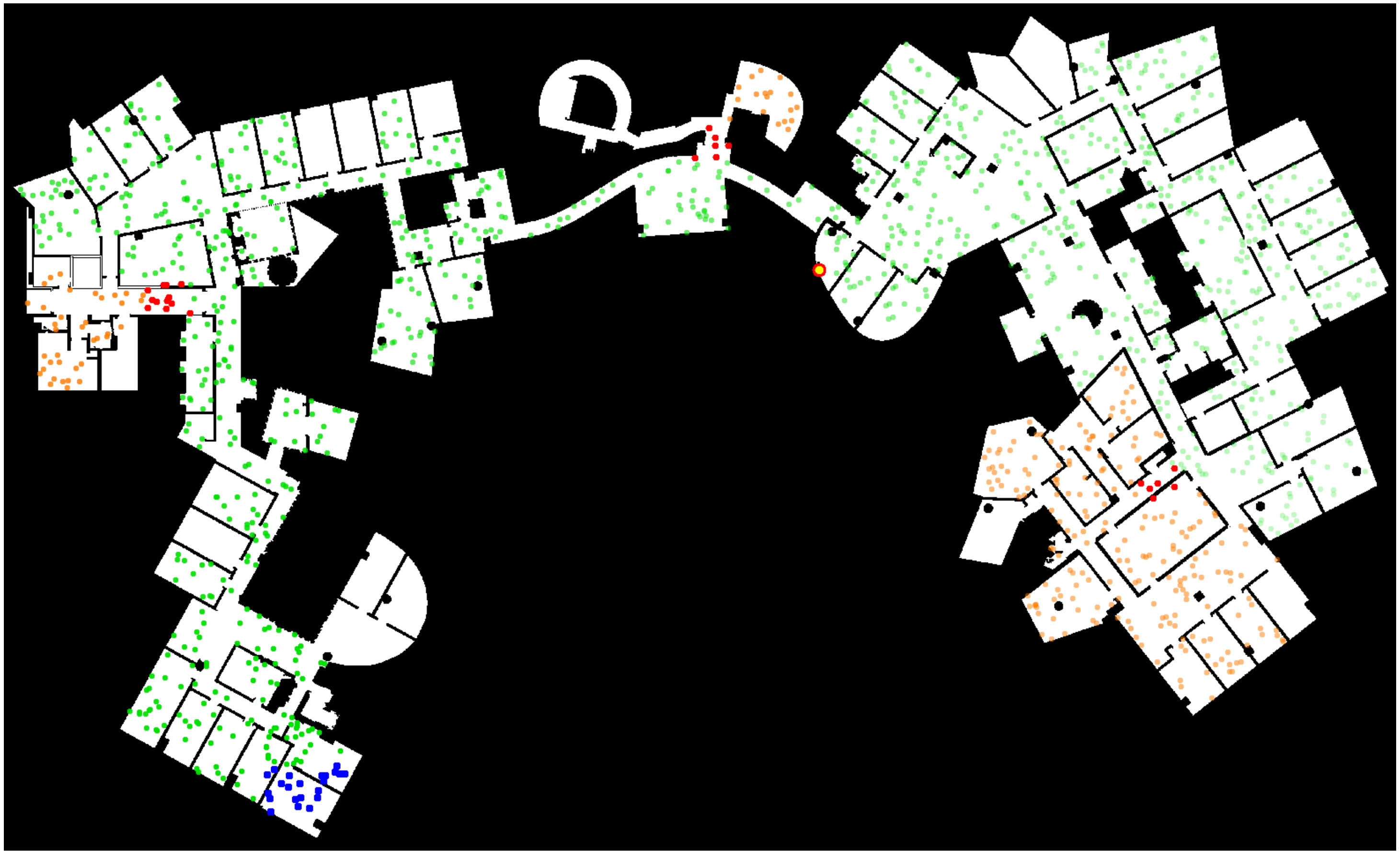
    G, except accumulation always integral, Boundary discarded

Example: Heterogeneous Density



GPI → path avoiding obstacles

Example: Building Evacuation



GPI → obstacle forecast: hazards (red) force orange areas to use alternate routes to blue safe zone

Summary

- Building-block algebras can be used to create aggregate APIs for implicitly adaptive programs
- 5-operator algebra gives implicit self-stabilization
- Field calculus is space-time universal
- GPI-calculus also resilient to device distribution

Summary and Open Problems

Jacob Beal, Mirko Viroli

Tutorial at 8th IEEE SASO
Imperial College London
September 2014

Raytheon
BBN Technologies

Summary

- Major technological trends are all driving towards a world filled with distributed systems
- Aggregate programming aims at rapid and reliable engineering of complex distributed systems
- Field calculus provides a universal theoretical foundation for aggregate programming
- Resilient systems design can be simplified by an emerging self-organization toolbox
- Functional composition allows modulation, predictable convergence

Lots of open problems…

Raytheon BBN Technologies

- Many field calculus results may be more generally applicable to all distributed systems
- Extent of basis set needed for various domains
- What implicit propeties are needed/possible for building block algebras and libraries?
- Extension to mobile devices, ongoing changes
- Prediction of approximation quality – “Nyquist”
- Higher order functions, first class functions
- Open environments and security

But we’ve already enough for many applications…

Bibliography

Raytheon BBN Technologies

- Viroli M, Damiani F, Beal J. A Calculus of Computational Fields. In: Canal C, Villari M, editors. Advances in Service-Oriented and Cloud Computing. vol. 393 of Communications in Computer and Information Sci. Springer Berlin Heidelberg; 2013. p. 114–128.
- Beal J, Bachrach J. Infrastructure for Engineered Emergence in Sensor/Actuator Networks. IEEE Intelligent Systems. 2006 March/April;21:10–19.
- Beal J, Viroli M, Damiani F. Towards a Unified Model of Spatial Computing. In: 7th Spatial Computing Workshop (SCW 2014). AAMAS 2014, Paris, France; 2014.
- Fernandez-Marquez J, Marzo Serugendo G, Montagna S, Viroli M, Arcos J. Description and composition of bio-inspired design patterns: a complete overview. Natural Computing. 2013;12(1):43–67.
- Viroli M, Damiani F. A Calculus of Self-stabilising Computational Fields. In: 16th Conference on Coordination Languages and Models (Coordination 2014); 2014. p. 163–178.
- Beal J, Viroli M. Building blocks for aggregate programming of self-organising applications. In: Workshop on Foundations of Complex Adaptive Systems (FOCAS); 2014.
- Beal J. A Basis Set of Operators for Space-Time Computations. In: Spatial Computing Workshop; 2010.
- Jacob Beal, Mirko Viroli, Danilo Pianini, and Ferruccio Damiani. Scale-independent computations in situated networks. *under review.*