Aggregate Programming

Jacob Beal

10th Swarm Intelligence Conf.
September 2016
A generative approach to safety

Restrict your development environment...

...to contain only resilient distributed systems.
Dealing with challenging platforms

Emerging Computational Platforms

Computational Field Programming Models

Inherently Resilient Distributed Systems

Pay a little efficiency, get a lot of programmability and resilience
Fundamentally different models

Isolate Systems
Extremely High FLOPs

High Dispersion
Moderate FLOPs

High Resolution Sense/Act
Abysmal FLOPs

How can we program aggregates adaptively & efficiently?
Are there commonalities that cross substrates?
Example: Services for Mass Events
Example: Managing Crowd Danger
Device-Centric Programming

- Explicit design of adaptation and communication
- Complex per-device multi-service application
- Intractable to ensure correct behavior
Aggregate Programming

- Implicit adaptation and communication
- Code each collective service independently
- Compose via scope and information flow
Aggregate Programming
Aggregate Programming Stack

Crowd Management
- dangerousDensity
- crowdTracking
- crowdWarning
- safeDispersal

Collective Behavior
- collectivePerception
- collectiveSummary
- managementRegions

Perception
- summarize
- average
- regionMax

Action
- distanceTo
- broadcast
- partition

State
- timer
- lowpass
- recentTrue

Resilient Coordination Operators

Field Calculus Constructs

Device Capabilities
- local functions
- communication
- state
- restriction

Developer APIs

Application Code

Aggregation Programming Stack
Example: Mesh-Network Cell Phones
Geometric Program: Channel

Source

Destination

(cf. Butera)
Computing with fields

```
source
  / \  
distance-to + distance-to
  |   |  
  |\  |   |\ 
  |  |   |  |
distance <= dilate
```

destination

width

```
+ 37
10
```
(Higher Order) Field Calculus

\[
e ::= \mathit{x} \mid \mathit{v} \mid (\mathit{e} \overline{e}) \mid (\mathit{rep}_x \mathit{w} \mathit{e}) \mid (\mathit{nbr} \mathit{e}) \mid (\mathit{if} \mathit{e} \mathit{e} \mathit{e})
\]
\[
\mathit{v} ::= \mathit{l} \mid \phi
\]
\[
\mathit{l} ::= \mathit{b} \mid \mathit{n} \mid \langle \mathit{l}, \mathit{l} \rangle \mid \mathit{e} \mid (\mathit{fun}(\overline{x}) \mathit{e})
\]
\[
\mathit{w} ::= \mathit{x} \mid \mathit{l}
\]
\[
\mathit{F} ::= (\mathit{def} \mathit{f} (\overline{x}) \mathit{e})
\]
\[
\mathit{P} ::= \overline{F} \mathit{e}
\]

**Literal Value**

**Variable**

**Local Ops**

**State**

**Communication**

**Domain Change**

expression

value

local value

variable or local value

user-defined function

program

[Virolì et al., '13, Damiani et al., '15]
Field Calculus is Space-Time Universal

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

[Beal, ’10; Beal et al., SCW 14]
Instantiation: 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)))
Instantiation: Protelis

- Java-hosted & integrated
- Java-like syntax
- Eclipse support

Architecture:

```python
def distanceTo(source) {
    rep(d <- Infinity) {
        mux (source) { 0 
        else { minHood(nbr{d} + nbrRange) }
    }
}
```
Using Protelis in your projects

http://protelis.org

Table of Contents:

- Why Protelis?
  - Further reading / references
- Developing with Protelis
- Contributing to Protelis
  - Current build status of Protelis
- History and Trivia

Why Protelis?
Aggregate Programming Stack

**Crowd Management**
- dangerousDensity
- crowdWarning
- crowdTracking
- safeDispersal

**Collective Behavior**
- collectivePerception
- collectiveSummary
- managementRegions
- ...

**Perception**
- summarize
- average
- regionMax
- ...

**Action**
- distanceTo
- broadcast
- partition
- ...

**State**
- timer
- lowpass
- recentTrue
- ...

**Resilient Coordination Operators**

**Field Calculus Constructs**

**Device Capabilities**
- sensors
- actuators
- local functions
- communication
- state
- restriction

**Application Code**
- Code
- Developer
- APIs
Example: Managing Crowd Danger
Example of a complex service

(def evacuate (zone coordinator alert)
  (let ((alerted
        (if zone
            (broadcast coordinator
            (collect-region
            (distance-to commander)
            alert))
          0)))))
(* alerted
  (follow-gradient
  (distance-to (not zone))))
Self-stabilization is hard to get right

Naïve geometry: when stationary, fine…
Self-stabilization is hard to get right

... but doesn’t correct properly for change.
Self-Stabilizing Building Blocks

Information spreading
Information collection
Short-term memory

Resilience by construction: all programs from these building blocks are also self-stabilizing!
All combinations are self-stabilizing!

Now program rapidly converges following changes
Aggregate Programming Stack
Applying building blocks:

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

*Since based on these building blocks, all programs built this way are self-stabilizing!*
(def crowd-tracking (p))
;; Consider only Fruin LoS E or F within last minute
(if (recently-true (> (density-est p) 1.08) 60)
;; Break into randomized “cells” and estimate danger of each
(+ 1 (dangerous-density (sparse-partition 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 (21 ops)
IF: 3   G: 11   C: 4   T: 3
Generalization: Self-Stabilizing Calculus

\[ e ::= x \mid v \mid (e \bar{e}) \mid (\text{rep } x \cdot w \cdot e) \mid \text{(nbr } e) \mid \text{(if } e \mid e \mid e)] \]
\[ v ::= \ell \mid \phi \]
\[ \ell ::= b \mid n \mid (\ell, \ell) \mid o \mid f \mid \text{(fun } \bar{x} \mid e) \]
\[ w ::= x \mid \ell \]
\[ F ::= (\text{def } f(\bar{x}) \cdot e) \]
\[ P ::= \overline{F} \cdot e \]

**Restrict field calculus by replacing** \( e \)** with** \( s \):

\[ s ::= \ell \mid x \mid (s \bar{s}) \mid (\text{nbr } s) \mid (\text{if } s \mid s \mid s) \]
\[ T(\text{rep } x \cdot w(\pi^{MB} \times \bar{s})) \]
\[ C(\text{rep } x \cdot w(\pi^{F} s^{A} \cdot (\text{nbr } (s \cdot x)) \cdot \bar{s})) \]
\[ G(\text{rep } x \cdot w(\pi (\pi' (\text{nbr } (\pi'' \times \bar{s''})) \cdot \bar{s'})) \cdot \bar{s})) \]
\[ \pi' \circ \pi = \pi^{MD}, \pi'' \circ \pi' = \pi^{MBP}, x \notin \text{FV}(s, \bar{s}, s^{A}) \]
Self-Stabilization $\rightarrow$ Substitution

Given functions $\lambda, \lambda'$ with same type, $\lambda$ is substitutable for $\lambda'$ iff for any self-stabilising list of expressions $e$, $(\lambda \ e)$ always self-stabilises to the same value as $(\lambda' \ e)$.
Optimization of Dynamics

"Building Block" Library → Self-Org. System Specification

Self-Org. System Specification → Decompose into Building Blocks

Decompose into Building Blocks → Minimal Resilient Implementation

Minimal Resilient Implementation → Substitution relations

Substitution relations → Substitution Library

Substitution Library → Optimize by substitution

Optimize by substitution → Optimized Implementation

[Virolí, Beal, Damiani & Pianini, SASO ‘15]
Naïve algorithm: when stationary, fine…
Optimization Example: Crowd Alert

... but dynamics can’t keep up with fast mobility.
Optimization Example: Crowd Alert

Optimized dynamics, however, work well.
Aggregate Programming Stack

Crowd Management
- dangerousDensity
- crowdWarning
- crowdTracking
- safeDispersal

Collective Behavior
- collectivePerception
- collectiveSummary
- managementRegions

Perception
- summarize
- average
- regionMax

Action
- distanceTo
- broadcast
- partition

State
- timer
- lowpass
- recentTrue

Application Code

Developer APIs

Resilient Coordination Operators

Field Calculus Constructs

Device Capabilities

built-ins

C

G

T

if

built-ins

nbr

rep

if

sensors

actuators

communication

state

restriction
def dangerousDensity(p, r) {
    let mr = managementRegions(r*2, () -> { nbrRange });
    let danger = average(mr, densityEst(p, r)) > 2.17 &&
                 summarize(mr, sum, 1 / p, 0) > 300;
    if(danger) { high } else { low }
}

def crowdTracking(p, r, t) {
    let crowdRgn = recentTrue(densityEst(p, r)>1.08, t);
    if(crowdRgn) { dangerousDensity(p, r) } else { none }
}

def crowdWarning(p, r, warn, t) {
    distanceTo(crowdTracking(p, r, t) == high) < warn
}

Dissemination of new versions

Pre-emptive modulation of priorities
Opportunistic Airborne Sensor Sharing

GIS-integrated adaptive mission planning

Highly effective sharing

Low computational cost

[Beal, Usbeck, Loyall & Metzler, in prep]
Dependency-Directed Recovery

Identify & shut down affected services

Non-dependent services still run

Dramatically better recovery time

Fewer services disrupted

[Clark, Beal, & Pal, SASO ‘15]
(def simple-sensor-actuator ()
  (let ((x (test-sensor)))
    (debug-1 x)
    (debug-2 (not x))))

[Beal et al., 2012]
Summary: Aggregate Methodology

Crowd Management
- dangerousDensity
- crowdTracking
- crowdWarning
- safeDispersal

Application Code

Collective Behavior
- collectivePerception
- collectiveSummary
- managementRegions

Developer APIs

Perception
- summarize
- average
- regionMax

Action
- distanceTo
- broadcast
- partition

State
- timer
- lowpass
- recentTrue

Resilient Coordination Operators

built-ins
- C
- G
- T
- if

Field Calculus Constructs

built-ins
- nbr
- rep
- if

Device Capabilities

sensors
- actuators
- communication
- state
- restriction

Self-Org. System Specification
- Decompose into Building Blocks
- Minimize by Substitution
- Optimize Implementation

"Building Block" Library
- Substitute relations

Substitution Library
- Minimal Resilient Implementation

“Building Block” Library
- “Building Block” Specification

Decompose(into(Building(Blocks(by(Substitution(relations(substitution(optimizedImplementation))))))))
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
Acknowledgements

Raytheon
BBN Technologies

• Shane Clark
• Partha Pal
• Kyle Usbeck

The University of Iowa

• Soura Dasgupta
• Raghu Mudumbai
• Amy Kumar

• Mirko Viroli
• Danilo Pianini
• Ferruccio Damiani