Problems of First-Class Functions over Space-Time

Jacob Beal

NEPLS, June 2012
What's a distributed first-class function?

Many different devices simultaneously...

- ... define
- ... invoke
- ... and execute
the same function, with interaction between them as needed.

**But what is “the same”?**
Motivation: Aggregate Programming

- P2P systems
- Decentralized cloud
- Sensor networks
- Robotics
- Massive multicore
- Bioengineering
- ...

System of Myriad Interconnected Computing Devices

Global-to-Local Compilation

Users / Applications
First Class Functions Checklist

- Function Calls
- Recursion
- Anonymous Functions
- Nested Functions
- Closures
- Higher-Order Functions
- Function-valued Fields
- Runtime Function Definition
Prior models limit generality

- Many-to-one (i.e. client-server)
- One-to-Many (i.e. parallel batch processing)
- Limited many-to-many:
  - Specialized p2p
  - Population protocols
  - Replicated state machine
  - etc...
Approach: Continuous Model

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

Approximable with:
- Discrete network of devices
- Signals transmitting state

Extreme case can specialize to all applications
(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)))
Continuous Space-Time Programs

Well-defined iff: each operator's inputs and outputs have same domain.

(except domain-change operators)

(+ 1 (test-sense))
Function Calls

(def inc (x) (+ x 1))
(inc (test-sense))
First Class Functions Checklist

- Function Calls
- Recursion
- Anonymous Functions
- Nested Functions
- Closures
- Higher-Order Functions
- Function-valued Fields
- Runtime Function Definition
Proto's Families of Primitives

Pointwise

Restriction

Feedback

Neighborhood

+ restrict

+ any-hood

delay

41

7

48

nbr
Focus: Restriction

Sub-manifold

Selector
Changing Field Domains

Well-defined iff output domain becomes subspace of input domain

Operator implicit in all domain-crossing uses of field.
(let ((z 3))
  (if (bool-sense) 2 (+ z 1)))
Branches and Function Calls

\[(\text{let ((z 3)) (if (bool-sense) 2 (inc z)))}\]

Manifold restriction → well-defined distributed function calls
(def factorial (x)
  (if (= x 0)
    1
    (* x (factorial (- x 1))))
)

Distributed Recursion
First Class Functions Checklist

✔ • Function Calls
✔ • Recursion
  • Anonymous Functions
  • Nested Functions
  • Closures
  • Higher-Order Functions
  • Function-valued Fields
  • Runtime Function Definition
Anonymous & Nested Functions

(let ((z 3))
  (if (bool-sense)
      2
      (inc z)))

(let ((z 3))
  (def inc (x) (+ x 1))
  (if (bool-sense)
      2
      (inc z)))

(let ((z 3))
  (def inc (x) (+ x 1))
  (let ((x 2))
    ((fun (x) (+ x 1)) z)))

purely syntactic solution...
First Class Functions Checklist

✔ • Function Calls
✔ • Recursion
✔ • Anonymous Functions
✔ • Nested Functions
  • Closures
  • Higher-Order Functions
  • Function-valued Fields
  • Runtime Function Definition
Closures via Restriction

(\text{let} \ ((y \ 2))\\  \ (\text{def inc-by} \ (x) \ (+ \ x \ y))\\  \ (\text{inc-by} \ (\text{test-sense})))

External references pass through \textit{restrict}.
Well-definedness moves to evaluation...
First Class Functions Checklist

- Function Calls
- Recursion
- Anonymous Functions
- Nested Functions
- Closures
- Higher-Order Functions
- Function-valued Fields
- Runtime Function Definition

Partial

This is as far as we're confident...

Notice that so far everything is handled at compile time
Aggregate vs. Local Definition
Problem #1: Well-Definedness

How can field domains within a function be expanded?
Problem #2: Function Equality

When should local function instances share their state?

```
(if (bool-sense)
  (fun (x) (any-hood (nbr x)))
  (fun (x) (all-hood (nbr x))))

(if (bool-sense)
  (fun (x) (any-hood (nbr x)))
  (fun (x) (any-hood (nbr x))))

(let ((x (test-sense)))
  (if (bool-sense)
    (fun () (any-hood (nbr x)))
    (fun () (any-hood (nbr x))))))

(let ((f (fun (x) (any-hood (nbr x))))
  (if (bool-sense) f f))
```
Problem #2: Function Equality

When should local function instances share their state?

(let ((x (vector-sense)))
  (map f x))
An unsatisfactory syntactic solution:

(procs (elt sources)
    ((var init evolve) ...)
    (same? run? &optional terminate?)
    . body)

This definition permits some intriguing possibilities, like functions with domains that self-overlap, but it's awkward.

There's got to be a more elegant way...
Conclusions & Invitation

- Aggregate programs need first-class functions.
- Restriction models enable function properties that can be resolved at compile-time.
- Runtime properties require expanding domains and testing function equality.

*We need more elegant ways of doing this!*

Get involved: http://proto.bbn.com
Acknowledgements

External Collaborators:

- Kyle Usbeck
- Brett Benyo
- Mirko Viroli
- Stefan Dulman
- Nikolaus Correll