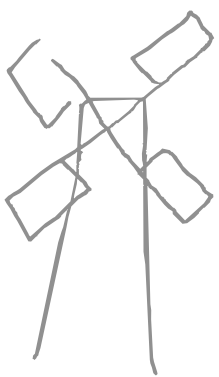
Finite Functional Programming

Michael Arntzenius · Berkeley

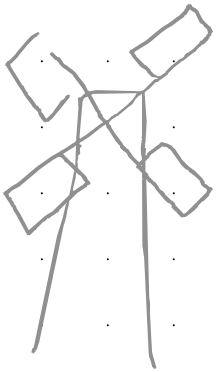
FLOPS 2026 · Tsukuba



use FANCY TYPES

to deconstruct LOGIC PROGRAMMING

into FUNCTIONAL PROGRAMMING



COMBINE Datalog — declarative, efficient

\+ Higher Order FP — modular, abstract

\+ Aggregation

\+ Tensor Algebra
(or Weighted Logic)

— expressive

USING Finitely supported maps,
pointed sets,
& point-preserving maps

LOGIC

```
follows(ada, bob).
follows(ada, charlie).
follows(bob, ada).
...


mutuals(X, Y) ←
    follows(X, Y),
    follows(Y, X).
```

LOGIC

```
follows(ada, bob).
follows(ada, charlie).
follows(bob, ada).
...

mutuals(X, Y) ←
    follows(X, Y),
    follows(Y, X).
```

ENUMERATES INPUTS THAT YIELD TRUE

?

FUNCTIONAL

```
follows : person → person → bool
follows x y =
    (x == "ada" and y == "bob")
or  (x == "ada" and y == "charlie")
or  (x == "bob" and y == "ada")
or  ...

mutuals : person → person → bool
mutuals x y =
    follows x y
and follows y x
```

INPUT → OUTPUT
BLACK BOX

LOGIC

costars(X, Y) ←
    stars(Film, X),
    stars(Film, Y).

LOGIC

```
costars(X, Y) ←
    stars(Film, X),
    stars(Film, Y).
```

FUNCTION

```
stars : film → person → bool

costars : person → person → bool
costars x y =
```

???

LOGIC

```
costars(X, Y) ←
    stars(Film, X),
    stars(Film, Y).
```

FUNCTION

```
stars : film → person → bool

costars : person → person → bool
costars x y =
  exists (λfilm. stars film x
                 and stars film y)
```

```
exists : (film → bool) → bool ??
```

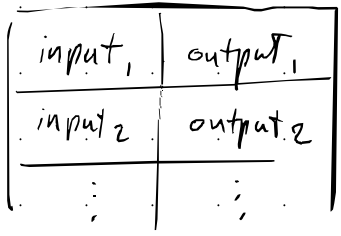
enumerate inputs that yield true?

we need FUNCTIONS

that can ENUMERATE THEIR SUPPORT !

IN THIS TALK: FINITELY SUPPORTED functions

IMPLEMENTED AS TABLES,



What is SUPPORT?

sets A, B

pointed sets P, Q sets with a "default element", $nil_P \in P$

eg. $nil_{bool} = false$

$$support\,(f : A \to P) \stackrel{def}{=} \{a \in A : f(a) \neq nil_P\}$$

$$A \xrightarrow{fs} P \stackrel{def}{=} \{f \in A \to P : support(f) \text{ is finite}\}$$

$$nil_{A \xrightarrow{fs} P} \stackrel{def}{=} \lambda x.\, nil_P$$

THE GOAL

a language that LOOKS functional
and supports "procedural" functions (incl. higher-order)
BUT ALSO has a type of "FINITE FUNCTIONS" that are

1. GUARANTEED finitely supported (via types)
2. REPRESENTED as TABLES
3. COMPUTED like DB QUERIES (w/ join algos)

EXAMPLES

```
costars : person -fs-> person -fs-> bool
costars x y =
    exists (λfilm. stars film x and stars film y)

-- # of films an actor has starred in
filmography : person -fs-> ℕ0
filmography actor = sum (λfilm. 1 when stars film actor)

-- shortest paths (via recursion — FUTURE WORK!)
distance : person -fs-> person -fs-> ℕ∞
distance x y =
    (1 when costars x y)
  min minimum (λz. distance x z + distance z y)
```

LOGIC

path(X,X) ← node(X).

path(X,Y) ← edge(X,Y).

path(X,Z) ← edge(X,Y), path(Y,Z).

WHY IS THIS FINITE?

LOGIC

$$path(X,X) \leftarrow node(X).$$
$$path(X,Y) \leftarrow edge(X,Y).$$
$$path(X,Z) \leftarrow edge(X,Y), path(Y,Z).$$

FUNCTION

$$path\ x\ z = (node\ x \ \underline{and}\ x = z)$$
$$\underline{or}\ edge\ x\ z$$
$$\underline{or}\ exists\ (\lambda y.\ edge\ x\ y\ \underline{and}\ path\ y\ z)$$

How can we check that this is FINITELY SUPPORTED?

I.E. How can we generalize GROUNDING to FUNCTIONAL PROGRAMS?

AND + OR

Given edge : $\mathbb{N} \xrightarrow{fs} \mathbb{N} \xrightarrow{fs}$ bool, which vars are ground in:

edge x y <u>and</u> edge y z ? x, y, z

edge x y <u>or</u> edge y z ? y

RULE:

t <u>and</u> u grounds x if either t or u does.

t <u>or</u> u grounds x if both t, u do.

BUT <u>WHY?</u>

nil and y = nil
x and nil = nil

⟹ (t and u) constrains x to a finite set if either t or u does!

IE grounds x

nil or nil = nil ⟹ (t or u) constrains x to a finite set if both t and u do!

CAN WE CAPTURE THIS WITH TYPES?

A, B sets $\qquad$ P, Q pointed sets, $nil_P \in P$

		INTENDED REPRESENTATION
$A \to B$	functions	procedural
$A \xrightarrow{fs} P$	finitely supported maps	tabled
$P \multimap Q$	point-preserving functions	procedural

$$P \multimap Q \stackrel{def}{=} \{ f : P \to Q \;:\; f(nil_P) = nil_Q \}$$

THEOREM: if $g : A \xrightarrow{fs} P$ and $f : P \multimap Q$ then $f \circ g : A \xrightarrow{fs} Q$ is finitely supported.

$P \,\&\, Q \overset{\text{def}}{=} P \times Q$

$\text{nil}_{P \,\&\, Q} \overset{\text{def}}{=} (\text{nil}_P, \text{nil}_Q)$

$\text{or} : \text{bool} \,\&\, \text{bool} \multimap \text{bool}$

BC $\text{or}(\text{false}, \text{false}) = \text{false}$

IE

$\text{or}(\text{nil}_{\text{bool}}, \text{nil}_{\text{bool}}) = \text{nil}_{\text{bool}}$

"DIRECT PRODUCT"

$P \otimes Q \overset{\text{def}}{=} P \times Q$ modulo

$\text{nil}_{P \otimes Q} \overset{\text{def}}{=} (\text{nil}_P, y) = (x, \text{nil}_Q)$

$\text{and} : \text{bool} \otimes \text{bool} \multimap \text{bool}$

BC $\text{and}(\text{false}, \_) = \text{false}$
$\text{and}(\_, \text{false}) = \text{false}$

IE

$\text{and}(\text{nil}, \_) = \text{nil}$
$\text{and}(\_, \text{nil}) = \text{nil}$

"SMASH PRODUCT"
or
TENSOR PRODUCT

EXISTS

$$\text{exists} : (A \xrightarrow{fs} \text{bool}) \overset{*}{\multimap} \text{bool}$$

exists (λn. n % 17 = n - 2) ✗ body doesn't ground n

exists (λfilm. stars film x) ✓ body grounds film

Grounds x because λfilm. stars film x does and exists preserves nil*!

LOGIC

$$path(X,X) \leftarrow node(X).$$
$$path(X,Y) \leftarrow edge(X,Y).$$
$$path(X,Z) \leftarrow edge(X,Y), path(Y,Z).$$

FUNCTION

$$path\ x\ z = (node\ x\ \underline{and}\ x = z)\ \underline{or}\ edge\ x\ z\ \underline{or}\ exists(\lambda y.\ edge\ x\ y\ \underline{and}\ path\ y\ z)$$

How do we check that this is FINITELY SUPPORTED?

Using POINT-PRESERVING MAPS!

$$T = R \circ S \iff Txz = \exists y.\, Rxy \wedge Syz$$

$$C = AB \iff Cik = \sum_j A_{ij} \cdot B_{jk}$$

$0 \cdot y = x \cdot 0 = 0$

$nil_{bool} = false$

$and : bool \otimes bool \multimap bool$

$exists : (A \xrightarrow{fs} bool) \multimap bool$

$nil_{\mathbb{R}_0} = 0$

$(\cdot) : \mathbb{R}_0 \otimes \mathbb{R}_0 \multimap \mathbb{R}_0$

$sum : (A \xrightarrow{fs} \mathbb{R}_0) \multimap \mathbb{R}_0$

Given $a : A \xrightarrow{fs} B \xrightarrow{fs} \mathbb{R}_0$, $b : B \xrightarrow{fs} C \xrightarrow{fs} \mathbb{R}_0$,

$c : A \xrightarrow{fs} C \xrightarrow{fs} \mathbb{R}_0$

$c = \lambda i.\, \lambda k.\, sum(\lambda j.\, a\, i\, j \cdot b\, j\, k)$

TENSOR ALGEBRA & AGGREGATION

$$\text{and} : \text{bool} \otimes \text{bool} \multimap \text{bool} \qquad \times : \mathbb{R}_0 \otimes \mathbb{R}_0 \multimap \mathbb{R}_0$$

$$\text{or} : \text{bool} \,\&\, \text{bool} \multimap \text{bool} \qquad + : \mathbb{R}_0 \,\&\, \mathbb{R}_0 \multimap \mathbb{R}_0$$

$$\text{exists} : (A \xrightarrow{fin} \text{bool}) \multimap \text{bool} \qquad \text{sum} : (A \xrightarrow{fin} \mathbb{R}_0) \multimap \mathbb{R}_0$$

$\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$

$\text{nil}_{\mathbb{N}_\infty} = \infty$

$$+_\infty : \mathbb{N}_\infty \otimes \mathbb{N}_\infty \multimap \mathbb{N}_\infty$$

$$\text{min} : \mathbb{N}_\infty \,\&\, \mathbb{N}_\infty \multimap \mathbb{N}_\infty$$

$$\text{minimum} : (A \xrightarrow{fin} \mathbb{N}_\infty) \multimap \mathbb{N}_\infty$$

ALSO USEFUL:

$$(=) : A \to A \xrightarrow{fin} \text{bool} \qquad \text{when} : P \otimes \text{bool} \multimap P$$

To check GROUNDEDNESS
(finite support)

we use point-preserving funs, $P \to Q$.

but how do we check
that a function
preserves nil?

GIVEN foo, bar : P ⊸ bool, which preserve nil?

✓ $\lambda x.x$: $P \multimap P$

✗ $\lambda x.$ true : $P \multimap$ bool

✓ $\lambda x.$ foo x and bar x : $P \multimap$ bool

~~LINEAR~~ exactly once ~~AFFINE~~ at most once

RELEVANT at least once

TYPE SYSTEM

A, B sets

P, Q pointed sets, $\mathrm{nil}_P \in P$

		INTENDED REPRESENTATION
$A \to B$	functions	procedural
$A \xrightarrow{fs} P$	finitely supported maps	tabled
$P \multimap Q$	point-preserving functions	procedural

check finite support by grounding, which depends on...

check point-preservation with relevance types.
(similar to linear types but with duplication)

λ: the ULTIMATE RELATION!

→ RELATIONS = FUNCTIONS + SUPPORT

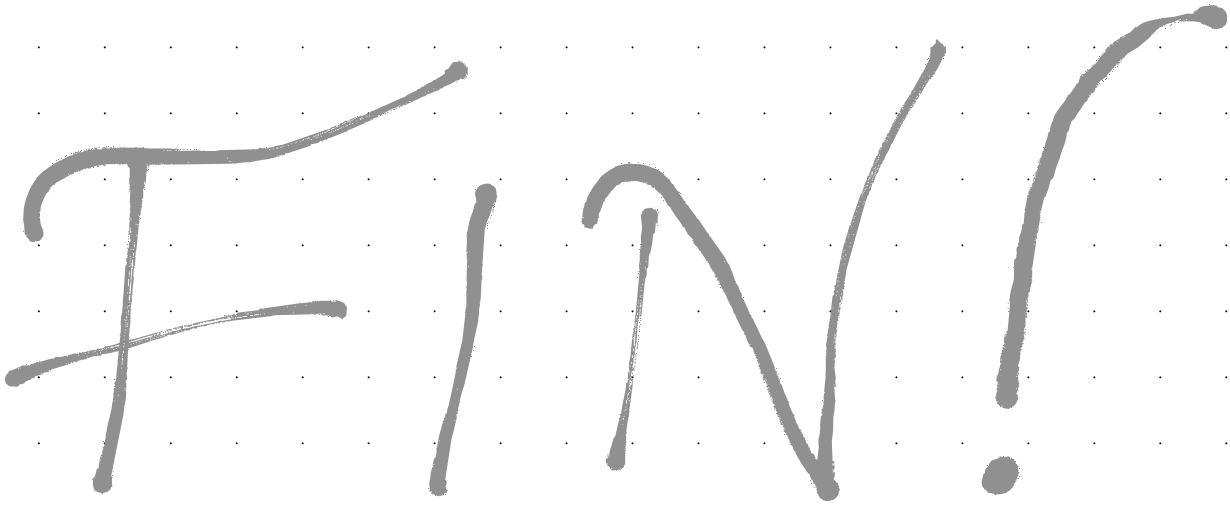
→ DATALOG = FINITE support ✓
PROLOG = ENUMERABLE, SYMBOLIC support ?

→ check finite support via POINTED SETS
& POINT-PRESERVING MAPS

→ naturally handles { TENSOR ALGEBRA
AGGREGATIONS

→ see paper for type system details
(it's complex but I kinda ♡ it)

DEMO TIME?


FIN!