Mode dependent dynamical systems and polynomial functors

David I. Spivak

March 5, 2020

Put Idris code on back board before starting.

time check: 5:05

I. Introduction

A. Motivation 3 mins

1. Want single formalism for many things:
   a. Bonding of atoms based on nearness in space
   b. When you pull too hard on something, it can break
   c. Opening and closing of eyes, extending of hands
   d. Busy and ready phases of workers in a bureaucracy
   e. NASA project limitation: planes connect differently at different times!
   f. When someone assembles a machine, their own outputs dictate the connection pattern of the machine's components.
   g. And of course Turing machines
2. Motto: wiring diagrams, where systems can choose who they wire to based on their internal states
   a. Want operadic, so that you can nest
3. Published something on this in 2015, but the CT was a bit ad hoc.
4. Recently found a really elegant formalism for it.

B. Relation to David Jaz's talks 4 mins

1. Recall
   a. Generalized lens setup $A\colon \mathcal{C}^{\mathrm{op}} \to \mathsf{Cat}$
   b. Apply a variant of the Grothendieck construction to get Lens_A
   c. Objects: (C, A), morphisms $(f\colon C \to C', f^\sharp\colon f^*A' \to A)$.
2. Today, $\mathcal{C} \coloneqq \mathsf{FinSet}$ or $\mathcal{C} \coloneqq \mathsf{Set}$ and $A \coloneqq \mathcal{C}/-$.

C. Plan 2 mins

1. Talk about the category Poly
2. Give an example of a machine that changes its inputs and outputs based on its state.

time check: 5:14

II. The category Poly

A. Definition of Poly — 5 mins

1. Full subcategory of FinSet $\to$ FinSet spanned by sums of representables.
   a. Objects: $P = \sum_{i:P(1)} y^{p_i}$
   b. Morphisms: $\mathsf{Poly}(P, Q) = \prod_{i:P(1)} Q(p_i)$.
   c. Morphisms between monomials: lens maps
   d. Example: $\mathsf{Poly}(By^A, y) = \mathsf{FinSet}(B, A)$.
2. $\mathsf{Poly}_{\mathsf{Set}}$ similar, replacing FinSet by Set.

time check: 5:19

B. Example: stream producers and transducers — 4 mins

1. Stream transducers
   a. Definition: An (A, B)-*stream transducer* consists of
      (1) A set S, "states"
      (2) Functions $r\colon S \to B$ and $u\colon S \to S$.
      (3) Initialized $s_0 : S$
      (4) $s_{n+1} = u(s_n)$, $b_n = r(s_n)$.
   b. As map of polynomials
      (1) By^A is called the *interface*
      (2) $Sy^S \to By^A$.
      (3) $y \to Sy^S$ for initialization
2. Special case: stream producer
   a. A B-*stream producer* is a $(1, B)$-stream transducer
   b. Map of polynomials $Sy^S \to By$.

time check: 5:23

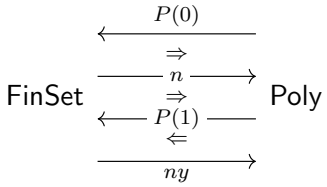
C. Properties of Poly

1. Has finite products and coproducts given by $0, +, 1, \times$ — 2 mins
2. Use case: Product of interfaces for stream transducers
   a. Example 1: — 3 mins
      (1) Interface product of $Sy^S \to Ay$ and $Sy^S \to By^2$.
      (2) Draw, use color.
   b. Special case: pause button — 3 mins

(1) Problem statement
   (a) A pause button for dynamical systems
   (b) Given a dynamical system, I want to be able to send a pause signal
   (c) Receiving non-pause inputs, it just goes about its usual business
   (d) Receiving pause input, it remains in the current state.
(2) Formalization
   (a) given $Sy^S \to P$
   (b) Always have $Sy^S \to y$
   (c) Multiply: $Sy^S \to yP$
   (d) Example: $Sy^S \to By$ becomes $sy^S \to By^2$.

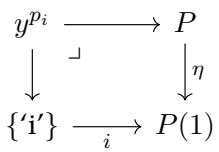
time check: 5:31

3. Poly has a string of adjoints 3 mins



and both functors out of FinSet are fully faithful (roundtrips on FinSet side are isos).

4. Poly has pullbacks. In particular for $i : P(1)$ have



so $P = \sum_{i:P(1)} P \times_{P(1)}$ 'i'.

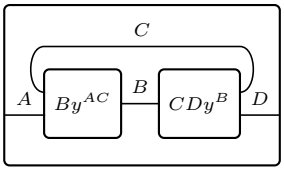
5. Composition monoidal product: $P \circ Q$. 1 min
6. Has $(\otimes, [-,-])$ adjunction 4 mins
   a. $\otimes$ is given by Day convolution of the Cartesian monoidal structure on FinSet.
      (1) On representables: $y^a \otimes y^b = y^{ab}$
      (2) Distributive
      (3) So $(3y^4 + 4y^2) \otimes (y^3 + 2) = 3y^{12} + 6 + 4y^6 + 8$.
      (4) From a bundle point of view: multiply bundles.
   b. $[P, Q] \coloneqq \prod_{i:P(1)} Q \circ (p_i y)$ 4 mins
      (1) Example: $[y^n, y] = ny$ and $[ny, y] = y^n$.
      (2) $\mathsf{Poly}(P \otimes A, Q) \cong \mathsf{Poly}(P, [A, Q])$.

D. Conclusion: it's a beautiful category!

time check: 5:43

III. Examples

A. Simple wiring diagrams 3 mins



$$\begin{aligned} By^{AC} \otimes CDy^B &\cong Dy^A \otimes BCy^{BC} \\ &\to Dy^A \otimes y \\ &\cong Dy^A \end{aligned}$$

1. Monomials By^A can be drawn as boxes with inputs A and outputs B.
2. The tensor of monomials is again a monomial $B_1y^{A_1} \otimes B_2y^{A_2} = B_1B_2y^{A_1A_2}$.
3. The operad of wiring diagrams (prisms) is a suboperad of the operad of monomials
4. Given dynamical systems $Sy^S \to By^{AC}$ and $Ty^T \to CDy^B$, we get a dynamical system

$$STy^{ST} \cong Sy^S \otimes Ty^T \to By^{AC} \otimes CDy^B \to Dy^A$$

time check: 5:46

B. Dynamical system with poly-interface:  9 mins

1. Adding natural numbers takes time no matter how you do it.
2. Here's a description of the old fashioned way, in terms of poly-interfaces.
   a. In ready state, orient for input  : accept two natural numbers (m, n) and go into work state.
   b. In work state where $m \neq 0$, orient as busy  : decrement the left number m, increment the right number n and stay in work state.
   c. When left number is $m = 0$, orient for output  : output the right number n and go to ready state.
3. In Idris:

$$S \coloneqq \mathbb{N}\mathbb{N} + 1 \qquad \texttt{Adder} : Sy^S \longrightarrow y^{\mathbb{N}\mathbb{N}} + y + \mathbb{N}y$$

```
S = Work Int Int | Ready          --states
O = Input | Busy | Output Int     --orientations

TS : S -> Type                    --move type for each state
TS _ = S                          --you can always move anywhere


I  : O -> Type                    --input type for each orientation

I Input      = Int Int            --during Input , take two Ints
I Busy       = ()                 --during Busy  , take no input
I (Output _) = ()                 --during Output, take no input

Adder : (s : S) -> (o : O, I o -> TS s)

Adder (Work 0 n)     = (Output n,  \ () -> Ready)
Adder (Work (m+1) n) = (Busy,      \ () -> Work m (n+1))
Adder Ready          = (Input, \ (m, n) -> Work mn)
```

time check: 5:55

IV. Conclusion 2 mins

A. Poly is a category with excellent formal properties

B. It naturally handles mode / context dependence

C. Next: need abstraction barriers for this.