Polynomials and the dynamics of data

David I. Spivak

TOPOS INSTITUTE

Seminario de categorías UNAM
2021 February 17

Outline

1 **Introduction**
- Personal history
- Plan

2 Theory

3 Applications

4 Speculations and questions

5 Conclusion

My personal history with math

I've always believed I could understand self, life, and world with math.

- We generally share experience and knowledge in "natural language".
- Is any of it inherently precluded from mathematical expression?

When I learned CT, I thought "this is where I can say it all."

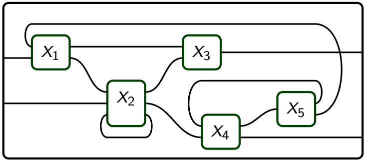
- It's a sublanguage of math that can talk about math.
- It's clean and principled and structural and expressive.

So I got to work trying to understand self, life, and world.

My personal history with ACT

What can we say about self, life, and world?

- I first assumed everything is information and communication.
  - Pretend our minds are information-storage devices.
  - How do we communicate with each other and with reality?
  - Understand everything in terms of databases and data migration!
    - (Categories, set-valued functors, parametric right adjoints.)
  - But interacting processes didn't seem to fit nicely.
- So then I assumed everything is interacting dynamical systems.
  - It's machines sending each other information, all the way down.



  - But should they really be wired the same way forever?

My personal history with Poly

Then one day I met **Poly** and fell in love.

- It captures dynamical systems and "rewiring diagrams".
- As a category it's exceptionally well-behaved.

The dynamics seemed to really be all about *comonoids* in **Poly**.

- Joachim Kock pointed me to R. Garner; I found his HoTTEST talk.
- Garner explained Ahman-Uustalu's result: "comonoids = categories"
- Garner also explained that bimodules = parametric right adjoints.

Suddenly everything I'd been working on for 13 years came together.

- I was overwhelmed by **Poly**'s elegance and capacity for application.
- It is extremely computational and hands-on...
- ...while displaying excellent formal properties.

Plan for today

Today's plan:

- Recall some basics of **Poly**;
- Show how **Poly** models dynamical systems and databases;
- Discuss some open questions and speculations; and
- Conclude with a brief summary.

Outline

1 Introduction

2 Theory
- **Poly** as a category
- Comonoids in **Poly**
- The framed bicategory $\mathbb{P}$

3 Applications

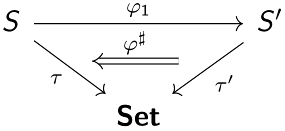
4 Speculations and questions

5 Conclusion

Poly for experts

What I'll call the category **Poly** has many names.

- The free completely distributive category on one object;
- The free coproduct completion of $\mathbf{Set}^{\mathrm{op}}$;
- The full subcategory of $[\mathbf{Set}, \mathbf{Set}]$ spanned by functors that preserve connected limits;
- The full subcategory of $[\mathbf{Set}, \mathbf{Set}]$ spanned by coproducts of repr'bles;
- The category of *typed sets* and colax maps between them.
  - Objects: pairs (S, τ), where $S \in \mathbf{Set}$ and $\tau \colon S \to \mathbf{Set}$.
  - Morphisms $(S, \tau) \xrightarrow{\varphi} (S', \tau')$: pairs $(\varphi_1, \varphi^\sharp)$, where



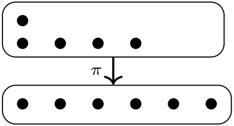
But let's make this easier.

What is a polynomial?

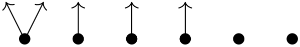
Algebraic

$$y^2 + 3y + 2$$

Bundle



Corolla forest



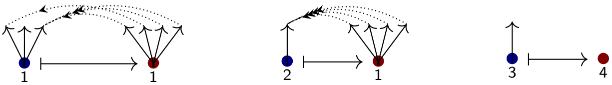
Interpretations:

- Each corolla in *p* is a position; its leaves are directions.
- Each corolla in *p* is a decision; its leaves are the options.

What is a morphism of polynomials?

Let $p := y^3 + 2y$ and $q := y^4 + y^2 + 2$

A morphism $p \xrightarrow{\varphi} q$ delegates each p-decision to a q-decision, passing back options:



Example: how to think of a map $y^2 + y^6 \to y^{52}$.

The category of polynomials

Easiest description: **Poly** = "sums of representables functors **Set** → **Set**".

- For any set S, let $y^S := \mathbf{Set}(S, -)$, the functor *represented* by S.
- Def: a polynomial is a sum $p = \sum_{i \in I} y^{p[i]}$ of representable functors.
- Def: a morphism of polynomials is a natural transformation.
- In **Poly**, $+$ is coproduct and $\times$ is product.

Notation

We said that a polynomial is a sum of representable functors

$$p \cong \sum_{i \in I} y^{p[i]}.$$

But note that $I \cong p(1)$. So we can write

$$p \cong \sum_{i \in p(1)} y^{p[i]}.$$

Composition monoidal structure $(\mathbf{Poly}, y, \triangleleft)$

The composite of two polynomial functors is again polynomial.

- Let's denote the composite of p and q by $p \triangleleft q$.
- Example: if $p := y^2$, $q := y + 1$, then $p \triangleleft q \cong y^2 + 2y + 1$.
- This is a monoidal structure, but not symmetric. $(q \triangleleft p \cong y^2 + 1)$
- The identity functor y is the unit: $p \triangleleft y \cong p \cong y \triangleleft p$.

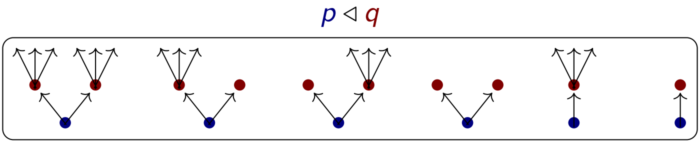
Why the we weird symbol $\triangleleft$ rather than $\circ$?

- We want to reserve $\circ$ for morphism composition.
- The notation $p \triangleleft q$ represents trees with p under q.

Composition given by stacking trees

Suppose $p := y^2 + y$ and $q := y^3 + 1$.

Draw the composite $p \triangleleft q$ by stacking q-trees on top of p-trees:



You can also read it as q feeding into p, which is how composition works.

Comonoids in $(\mathbf{Poly}, y, \triangleleft)$

In any monoidal category $(\mathcal{M}, I, \otimes)$, one can consider comonoids.

- A comonoid is a triple (m, ϵ, δ) satisfying certain rules, where
  - $m \in \mathcal{M}$ is an object, the *carrier*,
  - $\epsilon\colon m \to I$ is a map, the *counit*, and
  - $\delta\colon m \to m \otimes m$ is a map, the *comultiplication*.

In $(\mathbf{Poly}, y, \triangleleft)$, comonoids are exactly categories![1]

- If $\mathcal{C}$ is a category, the corresponding comonoid has carrier

$$\mathfrak{c} := \sum_{i \in \mathrm{Ob}(\mathcal{C})} y^{\mathfrak{c}[i]}$$

  where $\mathfrak{c}[i]$ is the set of morphisms in $\mathcal{C}$ that emanate from i.
- The counit $\epsilon\colon \mathfrak{c} \to y$ assigns to each object an identity.
- The comult $\delta\colon \mathfrak{c} \to \mathfrak{c} \triangleleft \mathfrak{c}$ assigns codomains and composites.

[1] Ahman-Uustalu. See my talk, https://www.youtube.com/watch?v=2mWnrgPIrIA

Comonoid maps are “cofunctors”

In **Poly**, comonoids are categories, but their morphisms aren’t functors.

- A comonoid morphism $\varphi\colon \mathcal{C} \nrightarrow \mathcal{D}$ is called a *cofunctor*.
- It includes a **Poly** map on carriers. For each object $i \in \mathfrak{c}(1)$, we get:
  - an object $j \coloneqq \varphi_1(i) \in \mathfrak{d}(1)$ and
  - for each emanating $f \in \mathfrak{d}[j]$, an emanating $\varphi_i^\sharp(f) \in \mathfrak{c}[i]$.

Example: what is a cofunctor $\mathcal{C} \overset{\varphi}{\nrightarrow} y^{\mathbb{N}}$?

- It is trivial on objects. On morphisms...
- ...it assigns an emanating morphism $\varphi_i^\sharp(1)$ to each object $i \in \mathfrak{c}(1)$.

“That’s not what you do with a category!”

- Cofunctors are kinda weird right? A whole new world to explore.
- A cofunctor $\mathcal{C} \nrightarrow y^{\mathbb{N}}$ is like a vector field on the category.
- This hints at applications, which are coming soon.

Bicomodules in $(\mathbf{Poly}, y, \triangleleft)$

Given comonoids $\mathscr{C}, \mathscr{D}$, a $(\mathscr{C}, \mathscr{D})$-bicomodule is another kind of map.

- It's a polynomial m, equipped with two maps

$$\mathfrak{c} \triangleleft m \longleftarrow m \longrightarrow m \triangleleft \mathfrak{d}$$

  each cohering naturally with the comonoid structure ϵ, δ.
- I denote this $(\mathscr{C}, \mathscr{D})$-bicomodule m like so:

$$\mathfrak{c} \triangleleft\!\!\xrightarrow{\;m\;}\!\!\triangleleft\; \mathfrak{d} \qquad \text{or} \qquad \mathscr{C} \triangleleft\!\!\xrightarrow{\;m\;}\!\!\triangleleft\; \mathscr{D}$$

  - The $\triangleleft$'s at the ends help me remember the how the maps go.
  - Maybe it looks like it's going the wrong way, but hold on.

Bicomodules are parametric right adjoints

Garner explained[2] that bicomodules $m \in {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$, which we've denoted

$$\mathcal{C} \overset{m}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathcal{D}$$

can be identified with parametric right adjoint functors (prafunctors)

$$\mathcal{D}\text{-}\mathbf{Set} \xrightarrow{M} \mathcal{C}\text{-}\mathbf{Set}.$$

- From this perspective the arrow points in the expected direction.
- Check: ${}_{\mathcal{C}}\mathbf{Mod}_0 \cong \mathcal{C}\text{-}\mathbf{Set}$.

Prafunctors $\mathcal{C} \triangleleft\!\!\longrightarrow\!\!\triangleleft \mathcal{D}$ generalize profunctors $\mathcal{C} \nrightarrow \mathcal{D}$:

- A profunctor $\mathcal{C} \nrightarrow \mathcal{D}$ is a functor $\mathcal{C} \to (\mathcal{D}\text{-}\mathbf{Set})^{\mathsf{op}}$
- A prafunctor $\mathcal{C} \triangleleft\!\!\longrightarrow\!\!\triangleleft \mathcal{D}$ is a functor $\mathcal{C} \to \mathbf{Coco}\big((\mathcal{D}\text{-}\mathbf{Set})^{\mathsf{op}}\big)$...
- ...where **Coco** is the free coproduct completion.

I'll explain how to think about it concretely when we get to applications.

[2] Garner's HoTTEST video, `https://www.youtube.com/watch?v=tW6HYnqn6eI`

The framed bicategory $\mathbb{P}$

Poly comonoids, cofunctors, and bicomodules form a framed bicategory $\mathbb{P}$.

- It's got a ton of structure, e.g. two monoidal structures, $+, \otimes$.
- Despite the last slide, it's actually not that hard to think about.

Here are some facts about ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$ for categories $\mathcal{C}, \mathcal{D}$.

- ${}_{\mathcal{D}}\mathbf{Mod}_{\mathbf{0}} \cong \mathcal{D}\text{-}\mathbf{Set}$, copresheaves on $\mathcal{D}$.
- ${}_{\mathbf{1}}\mathbf{Mod}_{\mathcal{D}} \cong \mathbf{Coco}\big((\mathcal{D}\text{-}\mathbf{Set})^{\mathsf{op}}\big)$.
- ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \cong \mathbf{Cat}(\mathcal{C}, {}_{\mathbf{1}}\mathbf{Mod}_{\mathcal{D}})$.

Outline

1 Introduction

2 Theory

3 Applications
- Interacting Moore machines
- Mode-dependence
- Databases

4 Speculations and questions

5 Conclusion

Moore machines

Definition

Given sets A, B, an (A, B)-*Moore machine* consists of:

- a set S, elements of which are called *states*,
- a function $r\colon S \to B$, called *readout*, and
- a function $u\colon S \times A \to S$, called *update*.

It is *initialized* if it is equipped also with

- an element $s_0 \in S$, called the *initial state*.

We refer to A as the *input set*, B as the *output set* of the Moore machine.

Dynamics: an (A, B)-Moore machine (S, r, u, s_0) is a "stream transducer":

- Given a list/stream $[a_0, a_1, \ldots]$ of A's...
- let $s_{n+1} := u(s_n, a_n)$ and $b_n := r(s_n)$.
- We thus have obtained a list/stream $[b_0, b_1, \ldots]$ of B's.

Moore machines as maps in Poly

We can understand Moore machines 

in terms of polynomials.

- An uninitialized Moore machine $r\colon S \to B$ and $u\colon S \times A \to S$ is:
  - A map of polynomials $Sy^S \to By^A$.
  - φ_1 is the readout and $\varphi^\sharp$ is the update.
- Add initialization by giving a map $y \to Sy^S$.

A *p-dynamical system* allows different input-sets at different positions.

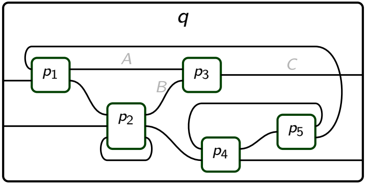
- For arbitrary $p \in \textbf{Poly}$ we can interpret a map $\varphi\colon Sy^S \to p$ as:
  - a readout: every state $s \in S$ gets a position $i := \varphi_1(s) \in p(1)$
  - an update: for every direction $d \in p[i]$, a next state $\varphi^\sharp_s(d) \in S$.
- Again, add initialization by giving a map $y \to Sy^S$.

Even more general: $Sy^S \nrightarrow \mathcal{C}$ for any category $\mathcal{C}$.

- For example, a map $Sy^S \to p$ can be identified with a cofunctor...
- ... $Sy^S \nrightarrow \mathfrak{c}_p$,, where $\mathfrak{c}_p$ is the *cofree comonoid* on p.

Wiring diagrams

We can have a bunch of dynamical systems interacting in an open system.



(φ)

Each box represents a monomial, e.g. $p_3 = Cy^{AB} \in \mathbf{Poly}$.

- The whole interaction, p_1 sending outputs to p_2 and p_3, etc....
- ... is captured by a map of polynomials $\varphi\colon p_1 \otimes \cdots \otimes p_5 \to q$. [3]
    - Given the positions (outputs) of each p_i, we get an output of q...
    - ... and when given an input of q, each p_i gets an input.

[3] Here $p \otimes p'$ just multiplies positions and directions,

$$p \otimes p' = \sum_{(i,i') \in p(1) \times p'(1)} y^{p[i] \times p'[i']}.$$

More general interaction

The whole picture above represents one morphism in **Poly**.

- Let's suppose the company chooses who it wires to; this is its *mode*.
- Then both suppliers have interface Wy for $W \in \mathbf{Set}$.
- Company interface is $2y^W$: two modes, each of which is W-input.
- The outer box is just y, i.e. a closed system.

So the picture represents a map $Wy \otimes Wy \otimes 2y^W \to y$.

- That's a map $2W^2 y^W \to y$.
- Equivalently, it's a function $2W^2 \to W$. Take it to be evaluation.
- In other words, the company's choice determines which $w \in W$ it receives.

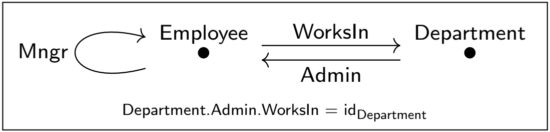
Other sorts of dynamical systems

Dynamical systems are usually defined as actions of a monoid T.

- Discrete: $\mathbb{N}$, reversible: $\mathbb{Z}$, real-time: $\mathbb{R}$.
- If T is a monoid and S is a set, a T-action on S is equivalently...
- ... a map $S \times T \to S$ satisfying two laws, which is equivalently...
- ... a cofunctor $Sy^S \nrightarrow y^T$, as in our general definition above.

Categorical databases

One view on databases is that they're basically just copresheaves.

$\mathcal{C} :=$ 

A functor $I\colon \mathcal{C} \to \mathbf{Set}$ (i.e. $\mathcal{C} \mathrel{\lhd\!\!\xrightarrow{I}\!\!\lhd} 0$) can be represented as follows:

Employee	**WorksIn**	**Mngr**
♡	P9	♡
T****	bLue	orca
orca	bLue	orca

Department	**Admin**
bLue	T****
P9	♡

But where's the data? What are the employees names, etc.?

More realistically, data should include *attributes* and look like this:

Employee	**FName**	**WorksIn**	**Mngr**
♡	Alan	P9	♡
T****	Dani	bLue	orca
orca	Sara	bLue	orca

Department	**DName**	**Secr**
bLue	Sales	T****
P9	IT	♡

- Assign a copresheaf $T\colon \operatorname{Ob}(\mathcal{C}) \to \mathbf{Set}$, e.g. $T(\text{Employee}) = \text{String}$.
- Using the canonical cofunctor $\mathcal{C} \nrightarrow \operatorname{Ob}(\mathcal{C})$, attributes are given by α:

Data migration

The framed bicategory structure of $\mathbb{P}$ is very useful in databases.

- We hinted at this in the last slide, adding attributes via a cofunctor.
- But so-called *data migration functors* are precisely prafunctors.

A prafunctor $\mathcal{C} \xleftarrow{P} \mathcal{D}$ in ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$ can be understood as follows.

- First, it's a functor $\mathcal{C} \to {}_{\mathbf{1}}\mathbf{Mod}_{\mathcal{D}}$, so what's that?
- We said it's a formal coproduct of formal limits in $\mathcal{D}$.
- A formal limit in $\mathcal{D}$ is called a *conjunctive query* on $\mathcal{D}$.
- So a prafunctor $\mathbf{1} \xleftarrow{Q} \mathcal{D}$ is a disjoint union of conjunctive queries.
- Let's call Q a duc-query on $\mathcal{D}$.

Example: if $\mathcal{D} = \left(\overset{\text{City}}{\bullet} \xrightarrow{\text{in}} \overset{\text{State}}{\bullet} \xleftarrow{\text{in}} \overset{\text{County}}{\bullet} \right)$, a duc-query might be...

$$(\text{City} \times_{\text{State}} \text{City}) + (\text{City} \times_{\text{State}} \text{County}) + (\text{County} \times_{\text{State}} \text{County})$$

A general bimodule $P \in {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$ is a $\mathcal{C}$-indexed duc-query on $\mathcal{D}$.

Outline

1. Introduction
2. Theory
3. Applications
4. **Speculations and questions**
   - Aggregation?
   - Metaphysical questions
5. Conclusion

Database aggregation

One of the most important uses of databases is aggregation.

- Setup: every employee is paid a salary and works in a department.
- Problem: assign each department the sum of its employees salaries.
- This is aggregation: not row-by-row; instead "rolling up a table".

I don't know of a nice ACT story for this anywhere.

- **Poly** loves databases and data migration.
- It's good at dynamics, e.g. "doing something" over and over.
- Isn't there some natural way to do aggregation?
- We'd start with a commutative monoid in the types; then what?

This is probably my current nomination for "#1 problem in ACT".

- It's a crucial step in understanding the nature of *summarizing*.
- In turn, summarizing is a huge metaphysical interest of mine.

A Poly-oriented view on metaphysics

I'll explain aspects of my current metaphysics using **Poly**.

- One's metaphysics is how they understand the fundamental principles.
- How does time work? What's up with identity? What is life?
- We can point at **Poly** while considering some of these things.

The following is just a play of forms, a submission I make for your review.

- Don't take this as a presentation of fact.
- Feel free to let me know what you think later.

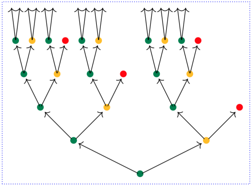
First a little more math: the cofree comonoid.

The cofree comonoid $\mathfrak{c}_p$

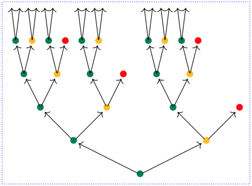
Comonoids in **Poly** are categories, so $\mathfrak{c}_p$ is a category; which one?

- It's actually free on a graph, but the graph is very interesting.
- The vertex-set $\mathfrak{c}_p(1)$ of the graph is the set of p-trees.
  - A p-tree is a possibly infinite tree t, where each node…
  - …is labeled by a position $i \in p(1)$ and has $p[i]$-many branches.
- For each vertex t, the set $\mathfrak{c}_p[t]$ of arrows emanating from t is…
- …the set of nodes n in tree t.
- Identity arrow = root node; codomain of n is the subtree at n.

Example object (tree) $t \in \mathfrak{c}_p$, where $p \cong 2y^2 + 1$:



Intuition from $\mathfrak{c}_p$



Suppose you (or the world) can be in $p(1)$-many positions, and...
...for each $i \in p(1)$, there are $p[i]$-many ways things might happen.

- Your character is how you respond in each such case.
- The character above always responds to left by turning green, etc.

The category of all "p-inhabiting characters" is $\mathfrak{c}_p$-**Set**, a topos.

- It's also the category of all dynamical systems with interface p.
- One can describe characters using the internal language of $\mathfrak{c}_p$-**Set**.
- We'll use an informal version to talk about experience.

What was, what's happening, and our character

Here are some assertions for your review:

- The past is irrevocably gone; it's always now.
- What we have of the past is what is left in the present.
  - This includes the layout of your surroundings.
  - It also includes the layout of your mind (memory).
  - The past—what was—is fossilized in the present layout.
  - We're continually consolidating experience; now, now, now.
- Imagine: all that remains of the past is the present position $i \in \mathfrak{c}_p(1)$.
  - What's happening now is the present direction $d \in p[i]$.
  - Imagine: our job is to compress the past into the present.
    - We try to remember something, we write it down, etc.
    - *Compression* because we encode both i and d in $\text{cod}(d)$.
  - Our character $X\colon \mathfrak{c}_p \to \mathbf{Set}$ is our compression scheme.
  - It's the type of responses we can have as things happen to us.

The lessons of history?

Imperative: compress the lessons of history to actualize ourselves.

- DNA compresses the lessons of who died, who survived, who thrived.
- History books, math books, culture: compress the lessons of history.
- But what's a lesson? What's worth compressing?
- Two senses of appreciation:
  - We pass on what we appreciate.
  - Appreciation of an asset is its growth.

How do you make math out of any of this?

- Polani's notion of Empowerment?
- Channel capacity between position now and direction in future.
- This may give a concrete notion of "lesson of history".

Factoring

Again for intuition only, imagine all of reality is embodied in p.

- Imagine you are a tensor factor, $p := p_1 \otimes p'$, ...
- ...where Ego = me = p_1, and Alter = environmnent=p'.
- Perhaps such factoring is a strategy for discerning the character of p?
- A map $p_1 \otimes p' \to y$ can be understood via standard cybernetics.
  - I present an unfolding situation for the environment, and...
  - ... the environment produces an unfolding situation for me.
  - We seem to pass constraints between characters in p_1 and p'.
  - But all of it is dictated by the character inhabiting p.

Is this sort of mereological breakdown actually useful? If so, what for?

Moving forward

The AI transition:

- Humans try to mimic intelligence they see in animals and people.
  - Example: "Computers" were originally people.
  - Turing explicitly designed machines to mimic their behavior.
  - We capture our understanding of life/intelligence in artifacts.
  - I'll call these artifacts "AI".
- AI can be run continuously at very fast rates.
- This has led to increasing complexity, already visible; more to come!

Mathematicians can enter the fray.

- If we say something in constructive math, technology can be formed.
- If what we say is elegant, the tech won't be ad-hoc.
- I prefer to be alongside elegant AI rather than ad-hoc AI.
- Mathematicians can join our historical moment and lead.

Poly is my entry point; you join our historical moment as you see fit.

Outline

1 Introduction

2 Theory

3 Applications

4 Speculations and questions

5 Conclusion
- Summary

Workshop on polynomial functors in March

Joachim Kock and I are organizing a **Poly** workshop.[4]

- Dates: March 15 – 19
- Speakers:

Thorsten Altenkirch	Steve Awodey
Michael Batanin	Bryce Clarke
Marcelo Fiore	Richard Garner
David Gepner	Helle Hvid Hansen
Rune Haugseng	Bart Jacobs
André Joyal	Fredrik Nordvall-Forsberg
Kristina Sojakova	David Spivak
Ross Street	Tarmo Uustalu

[4] `https://topos.site/p-func-2021-workshop/`

Summary

Poly is a category of remarkable abundance.

- It's completely combinatorial.
    - Calculations are concrete.
    - Much is already familiar, e.g. $(y+1)^2 \cong y^2 + 2y + 1$.
- It's theoretically beautiful.
    - Comonoids are categories.
    - Coalgebras are copresheaves.
- It's got a wide scope of applications.
    - Databases and data migration.
    - Dynamical systems and cellular automata.

A single setting for pursuing real philosophical and technological progress.

Thanks! Questions and comments welcome.