Polynomial functors II: Seven wonders of the composition product

David I. Spivak

MIT Categories Seminar
2020 May 28

Outline

1. **Introduction**
   - **Poly**: how do I love thee?
   - **Poly** and mode-dependent dynamical systems
2. Categorical virtues of Poly
3. Composition product in dynamical systems
4. Theoretical wonders of the $\circ$ monoidal structure
5. Conclusion

Poly: how do I love thee?

I’ve been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that

Poly: how do I love thee?

I've been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that everyone *ought* to give the object in question [his or her] approval

Poly: how do I love thee?

I've been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that everyone *ought* to give the object in question [his or her] approval and follow suit in describing it as beautiful....

Poly: how do I love thee?

I've been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that everyone *ought* to give the object in question [his or her] approval and follow suit in describing it as beautiful....
> We are suitors for agreement from everyone else....

Poly: how do I love thee?

I’ve been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that everyone *ought* to give the object in question [his or her] approval and follow suit in describing it as beautiful....
> We are suitors for agreement from everyone else....

As a lover of **Poly**, I feel drawn to tell you how great it is.

Poly: how do I love thee?

I've been totally enamored with **Poly** these past few months.

Kant, in *Critique of aesthetic judgment*:

> [A] person who describes something as beautiful insists that everyone *ought* to give the object in question [his or her] approval and follow suit in describing it as beautiful....
> We are suitors for agreement from everyone else....

As a lover of **Poly**, I feel drawn to tell you how great it is.

I hope you'll find it as fascinating as I do!

Some virtues of Poly

- **Poly** categorifies high-school math.
  - $y^2 + 2y + 1$ isn't intimidating; think inclusivity.
  - And $\mathbb{N}y^{\mathbb{Z}} + \texttt{String}\, y^{\texttt{Bool}}$ works the same basic way.

Some virtues of Poly

- **Poly** categorifies high-school math.
  - $y^2 + 2y + 1$ isn't intimidating; think inclusivity.
  - And $\mathbb{N}y^{\mathbb{Z}} + \texttt{String}\, y^{\texttt{Bool}}$ works the same basic way.
- **Poly** is easy to implement in a dependently-typed language, e.g. Idris.
  - It is exactly the theory of *containers*, if you've heard of that.
  - People can actually code up examples and run them.

Some virtues of Poly

- **Poly** categorifies high-school math.
  - $y^2 + 2y + 1$ isn't intimidating; think inclusivity.
  - And $\mathbb{N}y^{\mathbb{Z}} + \mathtt{String}\, y^{\mathtt{Bool}}$ works the same basic way.
- **Poly** is easy to implement in a dependently-typed language, e.g. Idris.
  - It is exactly the theory of *containers*, if you've heard of that.
  - People can actually code up examples and run them.
- **Poly** can be used to model mode-dependent dynamical systems.
  - I'll recall the basic ideas in the next slide.
  - Think: "wiring diagrams that change through time."
  - Atomic bonds or human communication channels form and break.

Some virtues of Poly

- **Poly** categorifies high-school math.
  - $y^2 + 2y + 1$ isn't intimidating; think inclusivity.
  - And $\mathbb{N}y^{\mathbb{Z}} + \texttt{String}\, y^{\texttt{Bool}}$ works the same basic way.
- **Poly** is easy to implement in a dependently-typed language, e.g. Idris.
  - It is exactly the theory of *containers*, if you've heard of that.
  - People can actually code up examples and run them.
- **Poly** can be used to model mode-dependent dynamical systems.
  - I'll recall the basic ideas in the next slide.
  - Think: "wiring diagrams that change through time."
  - Atomic bonds or human communication channels form and break.
- **Poly** has a laundry list of beautiful formal properties.
  - We'll discuss many of them today; some are quite surprising.
  - It's a rich categorical setting in which to work.

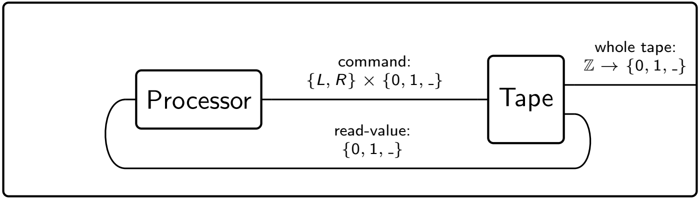
Some virtues of Poly

- **Poly** categorifies high-school math.
    - $y^2 + 2y + 1$ isn't intimidating; think inclusivity.
    - And $\mathbb{N}y^{\mathbb{Z}} + \texttt{String}\, y^{\texttt{Bool}}$ works the same basic way.
- **Poly** is easy to implement in a dependently-typed language, e.g. Idris.
    - It is exactly the theory of *containers*, if you've heard of that.
    - People can actually code up examples and run them.
- **Poly** can be used to model mode-dependent dynamical systems.
    - I'll recall the basic ideas in the next slide.
    - Think: "wiring diagrams that change through time."
    - Atomic bonds or human communication channels form and break.
- **Poly** has a laundry list of beautiful formal properties.
    - We'll discuss many of them today; some are quite surprising.
    - It's a rich categorical setting in which to work.

I talked about part of the story on March 5. There's a lot more to say.

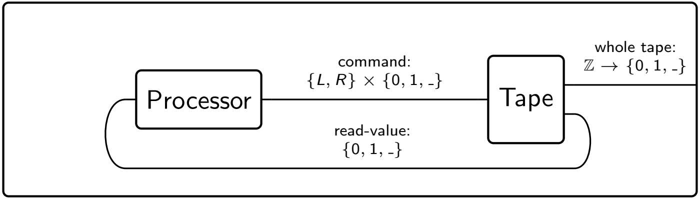
Poly and dynamical systems

A Turing machine includes two dyn'l systems with fixed interaction pattern:

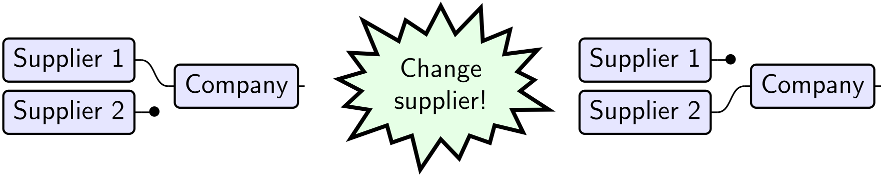


Poly and dynamical systems

A Turing machine includes two dyn'l systems with fixed interaction pattern:



Many systems in the real world change their interaction pattern.



We can model this—a company deciding to change its supplier—in **Poly**.

Previous talk vs. this talk

In the March 5 talk,[1] I discussed

- **Poly** as a category and its relationship to “generalized lenses”.

[1] All references will be given at the end of today’s talk.

Previous talk vs. this talk

In the March 5 talk,[1] I discussed

- **Poly** as a category and its relationship to “generalized lenses”.
- **Poly** as a model for mode-dependent dynamical systems.

[1] All references will be given at the end of today’s talk.

Previous talk vs. this talk

In the March 5 talk,[1] I discussed

- **Poly** as a category and its relationship to "generalized lenses".
- **Poly** as a model for mode-dependent dynamical systems.
  - In particular, I talked about monoidal structures $+, \times, \otimes$...
  - ... and how they model various ways of combining systems.

[1] All references will be given at the end of today's talk.

Previous talk vs. this talk

In the March 5 talk,[1] I discussed

- **Poly** as a category and its relationship to “generalized lenses”.
- **Poly** as a model for mode-dependent dynamical systems.
  - In particular, I talked about monoidal structures $+, \times, \otimes$...
  - ... and how they model various ways of combining systems.

But I avoided $\circ$ because frankly I didn’t know what it “did”.

In this talk I’ll recall the basics of the above, and then go into $\circ$.

- It’s got some real surprises up its proverbial sleeve.
- I’ll call them the *seven wonders of composition product*.

[1] All references will be given at the end of today’s talk.

Acknowledgments

I learned some of what I'll present here from (separate) conversations with Richard Garner and David Jaz Myers.

Outline

1 Introduction

2 **Categorical virtues of Poly**
- Recalling the category **Poly**
- An abundance of structure
- First wonder of composition product

3 Composition product in dynamical systems

4 Theoretical wonders of the $\circ$ monoidal structure

5 Conclusion

Poly has many different descriptions

Proposition

The following are equivalent:

- *The free distributive category (small products over small coproducts) on one object;*

Poly has many different descriptions

Proposition

The following are equivalent:

- *The free distributive category (small products over small coproducts) on one object;*
- *The full subcategory of* $[\mathbf{Set}, \mathbf{Set}]$ *spanned by functors that preserve connected limits;*

Poly has many different descriptions

Proposition

The following are equivalent:

- *The free distributive category (small products over small coproducts) on one object;*
- *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;*

Poly has many different descriptions

Proposition

The following are equivalent:

- *The free distributive category (small products over small coproducts) on one object;*
- *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 "generalized lens category" associated to the canonical self-indexing* $\mathbf{Set}/- \colon \mathbf{Set}^{\mathsf{op}} \to \mathbf{Cat}$ *of* $\mathbf{Set}$*;*

Poly has many different descriptions

Proposition

The following are equivalent:

- *The free distributive category (small products over small coproducts) on one object;*
- *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 "generalized lens category" associated to the canonical self-indexing* $\mathbf{Set}/- \colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ *of* $\mathbf{Set}$*;*
- *The category of typed sets* $\tau \colon S \to \mathbf{Set}$ *and colax maps between them*

What's a morphism $(S, \tau) \to (S', \tau')$? It's a choice of function $f \colon S \to S'$ and for each $s \in S$, a choice of function $f^\sharp_s \colon \tau'(fs) \to \tau(s)$.

$$\begin{tikzcd}
S \arrow[rr, "f"] \arrow[dr, "\tau"'] \arrow[rr, phantom, shift right=4, "\overset{f^\sharp}{\Longleftarrow}"] & & S' \arrow[dl, "\tau'"] \\
& \mathbf{Set} &
\end{tikzcd}$$

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ "represented" by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ “represented” by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.
- An arbitrary object $p \in \mathbf{Poly}$ is an arbitrary sum of representables

$$p = \sum_{i \in I} y^{p_i}$$

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ "represented" by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.
- An arbitrary object $p \in \mathbf{Poly}$ is an arbitrary sum of representables

$$p = \sum_{i \in I} y^{p_i}$$

  - As a functor, p sends a set X to the coproduct $\sum_{i \in I} X^{p_i}$.
  - Note that $I \cong p(1)$; we'll write $i \in p(1)$ from now on.

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ “represented” by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.
- An arbitrary object $p \in \mathbf{Poly}$ is an arbitrary sum of representables

$$p = \sum_{i \in I} y^{p_i}$$

  - As a functor, p sends a set X to the coproduct $\sum_{i \in I} X^{p_i}$.
  - Note that $I \cong p(1)$; we'll write $i \in p(1)$ from now on.
- The morphisms between polynomials are the natural transformations.

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ "represented" by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.
- An arbitrary object $p \in \mathbf{Poly}$ is an arbitrary sum of representables

$$p = \sum_{i \in I} y^{p_i}$$

  - As a functor, p sends a set X to the coproduct $\sum_{i \in I} X^{p_i}$.
  - Note that $I \cong p(1)$; we'll write $i \in p(1)$ from now on.
- The morphisms between polynomials are the natural transformations.
  - Hedges category of "bimorphic lenses" is equivalent to...
  - ...the full subcategory of **Poly** spanned by the monomials.
  - In those terms, a morphism $(f, f^\sharp)$ in **Poly** is a "view/update".

Today: represent objects in Poly as polynomials

- For each $S \in \mathbf{Set}$, there is a functor $\mathbf{Set}(S, -)$ "represented" by S.
- Denote it y^S for Yoneda, but also to look like a variable to a power.
  - As a functor, y^S sends a set X to the set of functions $S \to X$.
- An arbitrary object $p \in \mathbf{Poly}$ is an arbitrary sum of representables

$$p = \sum_{i \in I} y^{p_i}$$

  - As a functor, p sends a set X to the coproduct $\sum_{i \in I} X^{p_i}$.
  - Note that $I \cong p(1)$; we'll write $i \in p(1)$ from now on.
- The morphisms between polynomials are the natural transformations.
  - Hedges category of "bimorphic lenses" is equivalent to...
  - ...the full subcategory of **Poly** spanned by the monomials.
  - In those terms, a morphism $(f, f^\sharp)$ in **Poly** is a "view/update".

Nice thing: coproducts and products in **Poly** are standard $0, +$ and $1, \times$.

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

- Dirichlet product $\otimes$. Let $p = \sum_{i \in p(1)} y^{p_i}$ and $q = \sum_{j \in q(1)} y^{q_j}$; compare:

$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j}$$

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

- Dirichlet product $\otimes$. Let $p = \sum_{i \in p(1)} y^{p_i}$ and $q = \sum_{j \in q(1)} y^{q_j}$; compare:

$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j} \qquad \text{and} \qquad p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i q_j}.$$

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

- Dirichlet product $\otimes$. Let $p = \sum_{i \in p(1)} y^{p_i}$ and $q = \sum_{j \in q(1)} y^{q_j}$; compare:

$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j} \qquad \text{and} \qquad p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i q_j}.$$

  - $\otimes$ is a third symmetric monoidal product; its unit is y.
  - It distributes over $+$, i.e. $p \otimes (q_1 + q_2) \cong (p \otimes q_1) + (p \otimes q_2)$.

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

- Dirichlet product $\otimes$. Let $p = \sum_{i \in p(1)} y^{p_i}$ and $q = \sum_{j \in q(1)} y^{q_j}$; compare:

$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j} \qquad \text{and} \qquad p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i q_j}.$$

  - $\otimes$ is a third symmetric monoidal product; its unit is y.
  - It distributes over $+$, i.e. $p \otimes (q_1 + q_2) \cong (p \otimes q_1) + (p \otimes q_2)$.
  - Both $\times$ and $\otimes$ are closed monoidal structures.

$$q^p \cong \prod_{i \in p(1)} q \circ (p_i + y) \qquad \text{and} \qquad [p, q] \cong \prod_{i \in p(1)} q \circ (p_i y)$$

Four interacting monoidal structures

We've seen two monoidal structures on **Poly** $(+, \times)$; there are two more.

- Dirichlet product $\otimes$. Let $p = \sum_{i \in p(1)} y^{p_i}$ and $q = \sum_{j \in q(1)} y^{q_j}$; compare:

  $$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j} \qquad \text{and} \qquad p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i q_j}.$$

  - $\otimes$ is a third symmetric monoidal product; its unit is y.
  - It distributes over $+$, i.e. $p \otimes (q_1 + q_2) \cong (p \otimes q_1) + (p \otimes q_2)$.
  - Both $\times$ and $\otimes$ are closed monoidal structures.

  $$q^p \cong \prod_{i \in p(1)} q \circ (p_i + y) \qquad \text{and} \qquad [p, q] \cong \prod_{i \in p(1)} q \circ (p_i y)$$

- Composition product $\circ$; unit is y.
  - This monoidal structure is non-symmetric, $p \circ q \ncong q \circ p$.
  - We'll spend most of the remaining time discussing $\circ$.

A bit more structure to discuss

A bit more before we leave the formal structures of **Poly** and discuss $\circ$.

- Adjoints galore, (functors labeled by image of $A \in \mathbf{Set}$, $p \in \mathbf{Poly}$):

$$\begin{tikzcd}[column sep=huge]
\mathbf{Set} \arrow[r, shift left=3, "A" description] \arrow[r, shift right=9, "Ay"'] & \mathbf{Poly} \arrow[l, shift right=9, "p(0)"'] \arrow[l, shift right=3, "p(1)" description] \arrow[l, phantom, shift right=6, "\Rightarrow"] \arrow[l, phantom, "\Leftarrow"] \arrow[l, phantom, shift left=6, "\Rightarrow"]
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\mathbf{Set}^{\mathrm{op}} \arrow[r, shift left=3, "y^A"] & \mathbf{Poly} \arrow[l, shift left=3, "\Gamma p"] \arrow[l, phantom, "\Leftarrow"]
\end{tikzcd}
.$$

A bit more structure to discuss

A bit more before we leave the formal structures of **Poly** and discuss $\circ$.

- Adjoints galore, (functors labeled by image of $A \in \mathbf{Set}$, $p \in \mathbf{Poly}$):

$$\begin{tikzcd}[column sep=huge]
\mathbf{Set} \arrow[r, shift left=1, "A" description] \arrow[r, shift right=4, "Ay"'] \arrow[r, phantom, shift left=3.5, "\Rightarrow"] \arrow[r, phantom, shift right=0.5, "\Leftarrow"] \arrow[r, phantom, shift right=3, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift right=4, "p(0)"'] \arrow[l, shift left=1.5, "p(1)" description]
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\mathbf{Set}^{\mathrm{op}} \arrow[r, shift left=2, "y^A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=2, "\Gamma p"]
\end{tikzcd} .$$

- **Poly** $\to$ **Set** given by $p \mapsto p(1)$ is a monoidal $*$-bifibration.
  - Fiber over $I \in \mathbf{Set}$ is $(\mathbf{Set}/I)^{\mathrm{op}}$.

A bit more structure to discuss

A bit more before we leave the formal structures of **Poly** and discuss $\circ$.

- Adjoints galore, (functors labeled by image of $A \in \mathbf{Set}$, $p \in \mathbf{Poly}$):

$$\begin{tikzcd}[column sep=huge]
\mathbf{Set} \arrow[r, shift left=3, "A" description] \arrow[r, shift right=9, "Ay"'] \arrow[r, phantom, shift left=6, "\Rightarrow"] \arrow[r, phantom, "\Leftarrow"] \arrow[r, phantom, shift right=6, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift right=9, "p(0)"'] \arrow[l, shift left=3, "p(1)" description]
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\mathbf{Set}^{\mathrm{op}} \arrow[r, shift left=3, "y^A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=3, "\Gamma p"]
\end{tikzcd}
.$$

- **Poly** $\to$ **Set** given by $p \mapsto p(1)$ is a monoidal $*$-bifibration.
  - Fiber over $I \in \mathbf{Set}$ is $(\mathbf{Set}/I)^{\mathrm{op}}$.
  - For every $f\colon A \to B$ in **Set**, there's a pullback functor Δ_f.
  - Every such Δ_f has both a left and a right adjoint, Π_f and Σ_f.

A bit more structure to discuss

A bit more before we leave the formal structures of **Poly** and discuss $\circ$.

- Adjoints galore, (functors labeled by image of $A \in \mathbf{Set}$, $p \in \mathbf{Poly}$):

$$\begin{tikzcd} \mathbf{Set} \arrow[r, shift left=1ex, "A" description] \arrow[r, shift right=5ex, "Ay"'] & \mathbf{Poly} \arrow[l, shift right=5ex, "p(0)"'] \arrow[l, shift right=1ex, "p(1)" description] \end{tikzcd} \qquad \begin{tikzcd} \mathbf{Set}^{\text{op}} \arrow[r, shift left=1.5ex, "y^A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=1.5ex, "\Gamma p"] \end{tikzcd} .$$

- **Poly** $\to$ **Set** given by $p \mapsto p(1)$ is a monoidal $*$-bifibration.
  - Fiber over $I \in \mathbf{Set}$ is $(\mathbf{Set}/I)^{\text{op}}$.
  - For every $f \colon A \to B$ in **Set**, there's a pullback functor Δ_f.
  - Every such Δ_f has both a left and a right adjoint, Π_f and Σ_f.
  - Given $I \xleftarrow{e} E \xrightarrow{f} B \xrightarrow{g} J$, the usual poly'l fun'r is $\Delta_e \,⨾\, \Pi_f \,⨾\, \Sigma_g$.

A bit more structure to discuss

A bit more before we leave the formal structures of **Poly** and discuss $\circ$.

- Adjoints galore, (functors labeled by image of $A \in \mathbf{Set}$, $p \in \mathbf{Poly}$):

$$\begin{tikzcd}
\mathbf{Set} \arrow[r, shift left=3, "A" description] \arrow[r, shift right=9, "Ay"'] & \mathbf{Poly} \arrow[l, shift right=9, "p(0)"'] \arrow[l, shift left=3, "p(1)" description] \arrow[l, phantom, shift right=6, "\Rightarrow"] \arrow[l, phantom, "\Leftarrow"] \arrow[l, phantom, shift left=6, "\Rightarrow"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{Set}^{\mathrm{op}} \arrow[r, shift left=3, "y^A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=3, "\Gamma p"]
\end{tikzcd} .$$

- **Poly** $\to$ **Set** given by $p \mapsto p(1)$ is a monoidal $*$-bifibration.
  - Fiber over $I \in \mathbf{Set}$ is $(\mathbf{Set}/I)^{\mathrm{op}}$.
  - For every $f \colon A \to B$ in **Set**, there's a pullback functor Δ_f.
  - Every such Δ_f has both a left and a right adjoint, Π_f and Σ_f.
  - Given $I \xleftarrow{e} E \xrightarrow{f} B \xrightarrow{g} J$, the usual poly'l fun'r is $\Delta_e \fatsemi \Pi_f \fatsemi \Sigma_g$.
- **Poly** has a vertical/cartesian factorization system.
  - $(f, f^\sharp)$ *vertical* means f is iso; *cartesian* means each $f_i^\sharp$ is iso.
  - All four monoidal structures $+, \times, \otimes, \circ$ preserve cartesians.
  - All three symmetric ones $+, \times, \otimes$ preserve verticals.

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(This is the most mundane one; I wanted seven for numerological reasons.)

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(This is the most mundane one; I wanted seven for numerological reasons.)

- Commutation of $\circ$ on the left with $+, \times$: for any polynomials p_1, p_2, q,

$$(p_1 + p_2) \circ q \cong (p_1 \circ q) + (p_2 \circ q)$$
$$(p_1 \times p_2) \circ q \cong (p_1 \circ q) \times (p_2 \circ q)$$

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(This is the most mundane one; I wanted seven for numerological reasons.)

- Commutation of $\circ$ on the left with $+, \times$: for any polynomials p_1, p_2, q,

$$(p_1 + p_2) \circ q \cong (p_1 \circ q) + (p_2 \circ q)$$
$$(p_1 \times p_2) \circ q \cong (p_1 \circ q) \times (p_2 \circ q)$$

- Duoidal interaction of $\circ$ with $+, \otimes$: for any p_1, p_2, q_1, q_2, natural maps

$$(p_1 \circ p_2) + (q_1 \circ q_2) \to (p_1 + q_1) \circ (p_2 + q_2) \qquad \textit{(univ. prop. +)}$$
$$(p_1 \circ p_2) \otimes (q_1 \circ q_2) \to (p_1 \otimes q_1) \circ (p_2 \otimes q_2)$$

The "$+$"-one helps me remember the form of the "$\otimes$"-one.

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(This is the most mundane one; I wanted seven for numerological reasons.)

- Commutation of $\circ$ on the left with $+, \times$: for any polynomials p_1, p_2, q,

$$(p_1 + p_2) \circ q \cong (p_1 \circ q) + (p_2 \circ q)$$
$$(p_1 \times p_2) \circ q \cong (p_1 \circ q) \times (p_2 \circ q)$$

- Duoidal interaction of $\circ$ with $+, \otimes$: for any p_1, p_2, q_1, q_2, natural maps

$$(p_1 \circ p_2) + (q_1 \circ q_2) \to (p_1 + q_1) \circ (p_2 + q_2) \qquad \textit{(univ. prop. +)}$$
$$(p_1 \circ p_2) \otimes (q_1 \circ q_2) \to (p_1 \otimes q_1) \circ (p_2 \otimes q_2)$$

The "$+$"-one helps me remember the form of the "$\otimes$"-one.

Behold in wonder!

(1) First wonder of composition product

The composition product interacts really nicely with $+, \times, \otimes$.

(This is the most mundane one; I wanted seven for numerological reasons.)

- Commutation of $\circ$ on the left with $+, \times$: for any polynomials p_1, p_2, q,

$$(p_1 + p_2) \circ q \cong (p_1 \circ q) + (p_2 \circ q)$$
$$(p_1 \times p_2) \circ q \cong (p_1 \circ q) \times (p_2 \circ q)$$

- Duoidal interaction of $\circ$ with $+, \otimes$: for any p_1, p_2, q_1, q_2, natural maps

$$(p_1 \circ p_2) + (q_1 \circ q_2) \to (p_1 + q_1) \circ (p_2 + q_2) \qquad \textit{(univ. prop. +)}$$
$$(p_1 \circ p_2) \otimes (q_1 \circ q_2) \to (p_1 \otimes q_1) \circ (p_2 \otimes q_2)$$

The "$+$"-one helps me remember the form of the "$\otimes$"-one.

Behold in wonder! But seriously, **Poly** does have a great deal of structure.

The other wonders

Here are my “seven wonders” of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we’ve just seen;

The other wonders

Here are my "seven wonders" of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we've just seen;
2. Dynamical systems: relationship to *strategies*;

The other wonders

Here are my “seven wonders” of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we’ve just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;

The other wonders

Here are my "seven wonders" of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we've just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;
4. Dynamical systems and comonoids: *generalizing coalgebras*;

The other wonders

Here are my "seven wonders" of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we've just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;
4. Dynamical systems and comonoids: *generalizing coalgebras*;
5. Monoids (in a related category) are *operads* (Gambino-Kock);

The other wonders

Here are my "seven wonders" of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we've just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;
4. Dynamical systems and comonoids: *generalizing coalgebras*;
5. Monoids (in a related category) are *operads* (Gambino-Kock);
6. Comonoids are *categories* (Ahman-Uustalu);

The other wonders

Here are my "seven wonders" of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we've just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;
4. Dynamical systems and comonoids: *generalizing coalgebras*;
5. Monoids (in a related category) are *operads* (Gambino-Kock);
6. Comonoids are *categories* (Ahman-Uustalu);
7. Bimodules are *parametric right adjoints* (Garner).

The other wonders

Here are my “seven wonders” of the composition monoidal structure $\circ$:

1. Formal properties: interacts well with $+, \times, \otimes$, as we’ve just seen;
2. Dynamical systems: relationship to *strategies*;
3. Dynamical systems: *speeding up* the dynamics;
4. Dynamical systems and comonoids: *generalizing coalgebras*;
5. Monoids (in a related category) are *operads* (Gambino-Kock);
6. Comonoids are *categories* (Ahman-Uustalu);
7. Bimodules are *parametric right adjoints* (Garner).

The rest of the talk is explaining 2 – 7, in order.

Outline

1 Introduction

2 Categorical virtues of Poly

3 **Composition product in dynamical systems**
- Recalling polynomials in dynamics
- How $\circ$ relates to strategies
- How $\circ$ relates to speeding up dynamical systems
- How $\circ$ relates to generalizing coalgebras

4 Theoretical wonders of the $\circ$ monoidal structure

5 Conclusion

Imagining polynomials as arenas

For dynamical systems, think of a polynomial $p = \sum_{i \in p(1)} \mathcal{y}^{p_i}$ as an *arena*.

Imagining polynomials as arenas

For dynamical systems, think of a polynomial $p = \sum_{i \in p(1)} \mathcal{y}^{p_i}$ as an *arena*.

- The arena has various *positions*: "where you can be".
- At each position, there are various *distinctions*: "what you can see".

Imagining polynomials as arenas

For dynamical systems, think of a polynomial $p = \sum_{i \in p(1)} \mathcal{y}^{p_i}$ as an *arena*.

- The arena has various *positions*: “where you can be”.
- At each position, there are various *distinctions*: “what you can see”.
- The position set is $p(1)$, and for $i \in p(1)$ the distinction set is p_i.

Imagining polynomials as arenas

For dynamical systems, think of a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as an *arena*.

- The arena has various *positions*: "where you can be".
- At each position, there are various *distinctions*: "what you can see".
- The position set is $p(1)$, and for $i \in p(1)$ the distinction set is p_i.

So in the arena $8y^{\texttt{String}} + 8y^{\texttt{Bool}}$ there are 16 positions.

- At half, we'll see a string, at the other half we'll see a boolean.

Imagining polynomials as arenas

For dynamical systems, think of a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as an *arena*.

- The arena has various *positions*: "where you can be".
- At each position, there are various *distinctions*: "what you can see".
- The position set is $p(1)$, and for $i \in p(1)$ the distinction set is p_i.

So in the arena $8y^{\texttt{String}} + 8y^{\texttt{Bool}}$ there are 16 positions.

- At half, we'll see a string, at the other half we'll see a boolean.

Example: consider the polynomial Sy^S for a set S.

- A bit self-referential: there's a set of positions S.
- The distinctions available at $s \in S$ are always just elements of S.

How to think of $p \to q$ as a map of arenas

Given a morphism of polynomials $(f, f^\sharp)\colon p \to q$,

- We might think of arena p being an *internal world* inside of arena q.
- Each position $i \in p(1)$ is *externalized* as a position $f(i)$ in q.

How to think of $p \to q$ as a map of arenas

Given a morphism of polynomials $(f, f^\sharp)\colon p \to q$,

- We might think of arena p being an *internal world* inside of arena q.
- Each position $i \in p(1)$ is *externalized* as a position $f(i)$ in q.
- Each distinction $e \in q_{f(i)}$ is *internalized* as a distinction $f_i^\sharp(e) \in p(i)$.

How to think of $p \to q$ as a map of arenas

Given a morphism of polynomials $(f, f^\sharp)\colon p \to q$,

- We might think of arena p being an *internal world* inside of arena q.
- Each position $i \in p(1)$ is *externalized* as a position $f(i)$ in q.
- Each distinction $e \in q_{f(i)}$ is *internalized* as a distinction $f_i^\sharp(e) \in p(i)$.

For dynamics, call S a set of "states" and consider a morphism $S\mathcal{y}^S \to p$.

- Every internal state $s \in S$ is externalized as a position in the arena p.
- Every distinction available there is internalized to give a new state.

How to think of $p \to q$ as a map of arenas

Given a morphism of polynomials $(f, f^\sharp)\colon p \to q$,

- We might think of arena p being an *internal world* inside of arena q.
- Each position $i \in p(1)$ is *externalized* as a position $f(i)$ in q.
- Each distinction $e \in q_{f(i)}$ is *internalized* as a distinction $f_i^\sharp(e) \in p(i)$.

For dynamics, call S a set of "states" and consider a morphism $S\mathcal{y}^S \to p$.

- Every internal state $s \in S$ is externalized as a position in the arena p.
- Every distinction available there is internalized to give a new state.

Now let's talk about $\circ$ in this context.

A description of $p \circ q$

We can rewrite a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as follows:

$$p \cong \sum_{i \in p(1)} \prod_{d \in p_i} y.$$

- "A p-position and, for every p-distinction available there: a future."
- The term "future" indicates the functorial variable y, an unknown set.

A description of $p \circ q$

We can rewrite a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as follows:

$$p \cong \sum_{i \in p(1)} \prod_{d \in p_i} y.$$

- "A p-position and, for every p-distinction available there: a future."
- The term "future" indicates the functorial variable y, an unknown set.

So if p and q are polynomials, then $p \circ q$ has the formula:

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p_i} \sum_{j \in q(1)} \prod_{e \in q_j} y$$

A description of $p \circ q$

We can rewrite a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as follows:

$$p \cong \sum_{i \in p(1)} \prod_{d \in p_i} y.$$

- "A p-position and, for every p-distinction available there: a future."
- The term "future" indicates the functorial variable y, an unknown set.

So if p and q are polynomials, then $p \circ q$ has the formula:

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p_i} \sum_{j \in q(1)} \prod_{e \in q_j} y$$

- The sums are always over fixed sets, e.g. $p(1), q(1)$.
- The products are over sets that depend, e.g. on $i \in p(1)$ or $j \in q(1)$.
- Note that j can depend on i and d.

A description of $p \circ q$

We can rewrite a polynomial $p = \sum_{i \in p(1)} y^{p_i}$ as follows:

$$p \cong \sum_{i \in p(1)} \prod_{d \in p_i} y.$$

- "A p-position and, for every p-distinction available there: a future."
- The term "future" indicates the functorial variable y, an unknown set.

So if p and q are polynomials, then $p \circ q$ has the formula:

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p_i} \sum_{j \in q(1)} \prod_{e \in q_j} y$$

- The sums are always over fixed sets, e.g. $p(1), q(1)$.
- The products are over sets that depend, e.g. on $i \in p(1)$ or $j \in q(1)$.
- Note that j can depend on i and d.

"A p-position and, for every p-distinction available there: a q-position and for every q-distinction available there, a future."

(2) Strategies as composition-powers $p^{\circ n}$

It follows that for any polynomial $p \in \mathbf{Poly}$, we have

$$p^{\circ n} \cong \sum_{i_1 \in p(1)} \prod_{d_1 \in p_{i_1}} \sum_{i_2 \in p(1)} \prod_{d_2 \in p_{i_2}} \cdots \sum_{i_n \in p(1)} \prod_{d_n \in p_{i_n}} y$$

(2) Strategies as composition-powers $p^{\circ n}$

It follows that for any polynomial $p \in \textbf{Poly}$, we have

$$p^{\circ n} \cong \sum_{i_1 \in p(1)} \prod_{d_1 \in p_{i_1}} \sum_{i_2 \in p(1)} \prod_{d_2 \in p_{i_2}} \cdots \sum_{i_n \in p(1)} \prod_{d_n \in p_{i_n}} y$$

“A p-position and, for every p-distinction available there: a p-position and, for every p-distinction available there: . . . a p-position and, for every p-distinction available there: a future.

(2) Strategies as composition-powers $p^{\circ n}$

It follows that for any polynomial $p \in \textbf{Poly}$, we have

$$p^{\circ n} \cong \sum_{i_1 \in p(1)} \prod_{d_1 \in p_{i_1}} \sum_{i_2 \in p(1)} \prod_{d_2 \in p_{i_2}} \cdots \sum_{i_n \in p(1)} \prod_{d_n \in p_{i_n}} y$$

"A *p*-position and, for every *p*-distinction available there: a *p*-position and, for every *p*-distinction available there: . . . a *p*-position and, for every *p*-distinction available there: a future.

- Choosing a position in this arena is choosing a *length n-strategy*:
  - A choice of move, and whatever you get back, a choice of move...
  - ...*n* times.

(2) Strategies as composition-powers $p^{\circ n}$

It follows that for any polynomial $p \in \mathbf{Poly}$, we have

$$p^{\circ n} \cong \sum_{i_1 \in p(1)} \prod_{d_1 \in p_{i_1}} \sum_{i_2 \in p(1)} \prod_{d_2 \in p_{i_2}} \cdots \sum_{i_n \in p(1)} \prod_{d_n \in p_{i_n}} y$$

"A *p*-position and, for every *p*-distinction available there: a *p*-position and, for every *p*-distinction available there: . . . a *p*-position and, for every *p*-distinction available there: a future.

- Choosing a position in this arena is choosing a *length n-strategy*:
  - A choice of move, and whatever you get back, a choice of move...
  - ...n times.

This sort of combinatorial game is baked into the composition product.

Introduction to our hero: comonoids

For any set S, the polynomial functor Sy^S is a comonoid in $(\mathbf{Poly}, \circ, y)$

- It is well-known in functional programming.
    - They call Sy^S the *store* comonad (cousin to the *state* monad).
    - It comes from the adjunction $\begin{tikzcd} \mathbf{Set} \arrow[r, shift left, "- \times S"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Set} \arrow[l, shift left, "\_^S"] \end{tikzcd}$.

Introduction to our hero: comonoids

For any set S, the polynomial functor Sy^S is a comonoid in $(\textbf{Poly}, \circ, y)$

- It is well-known in functional programming.
    - They call Sy^S the *store* comonad (cousin to the *state* monad).
    - It comes from the adjunction $$\begin{tikzcd} \textbf{Set} \arrow[r, shift left=2, "- \times S"] \arrow[r, phantom, "\Rightarrow"] & \textbf{Set} \arrow[l, shift left=2, "\_^S"] \end{tikzcd}$$.
- Our use of Sy^S in dynamical systems stems from its comonoidality.

Introduction to our hero: comonoids

For any set S, the polynomial functor Sy^S is a comonoid in $(\mathbf{Poly}, \circ, y)$

- It is well-known in functional programming.
  - They call Sy^S the *store* comonad (cousin to the *state* monad).
  - It comes from the adjunction $\begin{tikzcd} \mathbf{Set} \arrow[r, shift left, "- \times S"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Set} \arrow[l, shift left, "\_^S"] \end{tikzcd}$.
- Our use of Sy^S in dynamical systems stems from its comonoidality.
  - ... the counit $\epsilon \colon Sy^S \to y$ says "stay where you are" and
  - ... the comultiplication δ says "make two moves in series".

Introduction to our hero: comonoids

For any set S, the polynomial functor Sy^S is a comonoid in $(\mathbf{Poly}, \circ, y)$

- It is well-known in functional programming.
  - They call Sy^S the *store* comonad (cousin to the *state* monad).
  - It comes from the adjunction $$\begin{tikzcd} \mathbf{Set} \arrow[r, shift left=2, "-\times S"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Set} \arrow[l, shift left=2, "\_^{S}"] \end{tikzcd}$$.
- Our use of Sy^S in dynamical systems stems from its comonoidality.
  - ... the counit $\epsilon\colon Sy^S \to y$ says "stay where you are" and
  - ... the comultiplication δ says "make two moves in series".

One might ask, "how should I think about comonoids in **Poly**, generally?"

- We'll discuss this soon; it's really a wonder.
- But first we'll show how we can use them in practice.

(3) Speeding up dynamical systems

Let (s, ϵ, δ) be a comonoid in $(\textbf{Poly}, \circ, y)$.

- We sometimes write $\delta\colon s \to s^{\circ n}$ for the $(n-1)$-fold iterate.
- For example, if $s := Sy^S$ then $\delta\colon s \to s^{\circ n}$ is "move n times", as above.

(3) Speeding up dynamical systems

Let (s, ϵ, δ) be a comonoid in $(\mathbf{Poly}, \circ, y)$.

- We sometimes write $\delta\colon s \to s^{\circ n}$ for the $(n-1)$-fold iterate.
- For example, if $s := Sy^S$ then $\delta\colon s \to s^{\circ n}$ is “move n times”, as above.

Given any $n \in \mathbb{N}$ and morphism $f\colon s \to p$ out of this comonoid, we get

- $f^{\circ n}\colon s^{\circ n} \to p^{\circ n}$ by functoriality of the monoidal product $\circ$.
- Precompose it with $\delta\colon s \to s^{\circ n}$ to obtain

$$s \xrightarrow{\delta} s^{\circ n} \xrightarrow{f^{\circ n}} p^{\circ n}$$

(3) Speeding up dynamical systems

Let (s, ϵ, δ) be a comonoid in $(\mathbf{Poly}, \circ, y)$.

- We sometimes write $\delta \colon s \to s^{\circ n}$ for the $(n-1)$-fold iterate.
- For example, if $s := Sy^S$ then $\delta \colon s \to s^{\circ n}$ is "move n times", as above.

Given any $n \in \mathbb{N}$ and morphism $f \colon s \to p$ out of this comonoid, we get

- $f^{\circ n} \colon s^{\circ n} \to p^{\circ n}$ by functoriality of the monoidal product $\circ$.
- Precompose it with $\delta \colon s \to s^{\circ n}$ to obtain

$$s \xrightarrow{\delta} s^{\circ n} \xrightarrow{f^{\circ n}} p^{\circ n}$$

This is a new dynamical system with the same states but faster.

- In every moment, s outputs an entire length-n strategy.

(3) Speeding up dynamical systems

Let (s, ϵ, δ) be a comonoid in $(\textbf{Poly}, \circ, y)$.

- We sometimes write $\delta\colon s \to s^{\circ n}$ for the $(n-1)$-fold iterate.
- For example, if $s := Sy^S$ then $\delta\colon s \to s^{\circ n}$ is "move n times", as above.

Given any $n \in \mathbb{N}$ and morphism $f\colon s \to p$ out of this comonoid, we get

- $f^{\circ n}\colon s^{\circ n} \to p^{\circ n}$ by functoriality of the monoidal product $\circ$.
- Precompose it with $\delta\colon s \to s^{\circ n}$ to obtain

$$s \xrightarrow{\delta} s^{\circ n} \xrightarrow{f^{\circ n}} p^{\circ n}$$

This is a new dynamical system with the same states but faster.

- In every moment, s outputs an entire length-n strategy.
- A p-position, and for every p-distinction, a choice of p-position, etc.

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x, a), & x(t) \in \mathbb{R}^d, \quad a(t) &\in \mathbb{R}^m \\ b &= g(x) & b(t) &\in \mathbb{R}^n. \end{aligned}$$

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x, a), & x(t) \in \mathbb{R}^d, \quad a(t) \in \mathbb{R}^m \\ b &= g(x) & b(t) \in \mathbb{R}^n. \end{aligned}$$

Let $X := \mathbb{R}^d$, $A := \mathbb{R}^m$, $B := \mathbb{R}^n$, and consider the polynomial map

$$(g, x{+}f) \colon X\mathcal{y}^X \to B\mathcal{y}^A$$

- Given $x \in \mathbb{R}^d$, get $b := g(x)$, and
- Given $x \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$, get an updated $x + f(x, a) \in X$.

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x,a), & x(t) \in \mathbb{R}^d, \quad a(t) \in \mathbb{R}^m \\ b &= g(x) & b(t) \in \mathbb{R}^n. \end{aligned}$$

Let $X := \mathbb{R}^d$, $A := \mathbb{R}^m$, $B := \mathbb{R}^n$, and consider the polynomial map

$$(g, x{+}f)\colon X\mathcal{y}^X \to B\mathcal{y}^A$$

- Given $x \in \mathbb{R}^d$, get $b := g(x)$, and
- Given $x \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$, get an updated $x + f(x,a) \in X$.
- So the dynamics of $(g, x{+}f)$ is doing Euler's method, time-step=1.

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x, a), & x(t) \in \mathbb{R}^d, \quad a(t) \in \mathbb{R}^m \\ b &= g(x) & b(t) \in \mathbb{R}^n. \end{aligned}$$

Let $X := \mathbb{R}^d$, $A := \mathbb{R}^m$, $B := \mathbb{R}^n$, and consider the polynomial map

$$(g, x{+}f) \colon X\mathcal{y}^X \to B\mathcal{y}^A$$

- Given $x \in \mathbb{R}^d$, get $b := g(x)$, and
- Given $x \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$, get an updated $x + f(x, a) \in X$.
- So the dynamics of $(g, x{+}f)$ is doing Euler's method, time-step$=1$.

Now let $f_n(x, a) := f(x, a)/n$ and consider $(g, x{+}f_n) \colon X\mathcal{y}^X \to B\mathcal{y}^A$.

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x, a), & x(t) \in \mathbb{R}^d, \quad a(t) \in \mathbb{R}^m \\ b &= g(x) & b(t) \in \mathbb{R}^n. \end{aligned}$$

Let $X := \mathbb{R}^d$, $A := \mathbb{R}^m$, $B := \mathbb{R}^n$, and consider the polynomial map

$$(g, x{+}f) \colon X\mathcal{y}^X \to B\mathcal{y}^A$$

- Given $x \in \mathbb{R}^d$, get $b := g(x)$, and
- Given $x \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$, get an updated $x + f(x, a) \in X$.
- So the dynamics of $(g, x{+}f)$ is doing Euler's method, time-step$=1$.

Now let $f_n(x, a) := f(x, a)/n$ and consider $(g, x{+}f_n) \colon X\mathcal{y}^X \to B\mathcal{y}^A$.

- By the work above we can speed it up to get $X\mathcal{y}^X \to (B\mathcal{y}^A)^{\circ n}$.
- In a single instant, it makes n-many Euler updates, time-step$=1/n$.

Example: differential equations

Consider a system of differential equations:

$$\begin{aligned} \dot{x} &= f(x, a), & x(t) \in \mathbb{R}^d, \quad a(t) \in \mathbb{R}^m \\ b &= g(x) & b(t) \in \mathbb{R}^n. \end{aligned}$$

Let $X := \mathbb{R}^d$, $A := \mathbb{R}^m$, $B := \mathbb{R}^n$, and consider the polynomial map

$$(g, x{+}f) \colon Xy^X \to By^A$$

- Given $x \in \mathbb{R}^d$, get $b := g(x)$, and
- Given $x \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$, get an updated $x + f(x, a) \in X$.
- So the dynamics of $(g, x{+}f)$ is doing Euler's method, time-step=1.

Now let $f_n(x, a) := f(x, a)/n$ and consider $(g, x{+}f_n) \colon Xy^X \to By^A$.

- By the work above we can speed it up to get $Xy^X \to (By^A)^{\circ n}$.
- In a single instant, it makes n-many Euler updates, time-step=$1/n$.
- As $n \to \infty$ (categorically??), one recovers the original ODE.

(4) Generalizing [Lambek] coalgebras

For $p \in \mathbf{Poly}$, a p-coalgebra is a set S and a function $f\colon S \to p(S)$.

(4) Generalizing [Lambek] coalgebras

For $p \in \mathbf{Poly}$, a p-coalgebra is a set S and a function $f\colon S \to p(S)$.

- But that's the same thing as a map $S\mathcal{y}^S \to p$!
- We can think about the coalgebra without having to leave **Poly**.

(4) Generalizing [Lambek] coalgebras

For $p \in$ **Poly**, a p-coalgebra is a set S and a function $f \colon S \to p(S)$.

- But that's the same thing as a map $Sy^S \to p$!
- We can think about the coalgebra without having to leave **Poly**.

What makes it work is that Sy^S is a comonoid.

- The map $Sy^S \to p$ induces a comonoid map $Sy^S \to$ **Cofree**(p).
- Here's a formula for **Cofree**(p) as a directed limit:

$$1 \xleftarrow{!} yP(1) \xleftarrow{yP(!)} yP(yP(1)) \xleftarrow{yP(yP(!))} yP(yP(yP(1))) \leftarrow \cdots$$

(4) Generalizing [Lambek] coalgebras

For $p \in \mathbf{Poly}$, a p-coalgebra is a set S and a function $f\colon S \to p(S)$.

- But that's the same thing as a map $Sy^S \to p$!
- We can think about the coalgebra without having to leave **Poly**.

What makes it work is that Sy^S is a comonoid.

- The map $Sy^S \to p$ induces a comonoid map $Sy^S \to \mathbf{Cofree}(p)$.
- Here's a formula for $\mathbf{Cofree}(p)$ as a directed limit:

$$1 \xleftarrow{!} yP(1) \xleftarrow{yP(!)} yP(yP(1)) \xleftarrow{yP(yP(!))} yP(yP(yP(1))) \leftarrow \cdots$$

- $\mathbf{Cofree}(p)(1)$ returns the usual formula for p's terminal coalgebra.

(4) Generalizing [Lambek] coalgebras

For $p \in \textbf{Poly}$, a p-coalgebra is a set S and a function $f \colon S \to p(S)$.

- But that's the same thing as a map $Sy^S \to p$!
- We can think about the coalgebra without having to leave **Poly**.

What makes it work is that Sy^S is a comonoid.

- The map $Sy^S \to p$ induces a comonoid map $Sy^S \to \textbf{Cofree}(p)$.
- Here's a formula for $\textbf{Cofree}(p)$ as a directed limit:

$$1 \xleftarrow{!} yP(1) \xleftarrow{yP(!)} yP(yP(1)) \xleftarrow{yP(yP(!))} yP(yP(yP(1))) \leftarrow \cdots$$

- $\textbf{Cofree}(p)(1)$ returns the usual formula for p's terminal coalgebra.

For any comonoid s and map $s \to p$, get $s \to \textbf{Cofree}(p)$.

- We'll see soon that the comonoid acts as a historical recorder.
- The comonoid $s = Sy^S$ says "to me, a history is just its endpoints."
- We'll see what it means that more general comonoids record history.

Outline

1 Introduction

2 Categorical virtues of Poly

3 Composition product in dynamical systems

4 **Theoretical wonders of the $\circ$ monoidal structure**
- Operads as monoids
- Categories as comonoids
- Bimodules are parametric right adjoints

5 Conclusion

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f_i^\sharp$ is an iso for each $i \in p(1)$.

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f_i^\sharp$ is an iso for each $i \in p(1)$.

A special case of a result by Gambino and Kock says the following.

- Let $\mathbf{Poly}^{\mathbf{cart}}$ denote the category of polynomials and cartesian maps.

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f_i^\sharp$ is an iso for each $i \in p(1)$.

A special case of a result by Gambino and Kock says the following.

- Let $\mathbf{Poly}^{\mathbf{cart}}$ denote the category of polynomials and cartesian maps.
- The category of monoids in $\mathbf{Poly}^{\mathbf{cart}}$ is equivalent to...
- ...the category of (one-object, nonsymmetric, infinitary) operads.
  - Example: List has a monoid structure in $\mathbf{Poly}^{\mathbf{cart}}$.
  - It corresponds to the “associative operad”; one op’n of each arity.
- If you want the cat’y of finitary operads, use monoids over List.

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f_i^\sharp$ is an iso for each $i \in p(1)$.

A special case of a result by Gambino and Kock says the following.

- Let $\mathbf{Poly}^{\mathbf{cart}}$ denote the category of polynomials and cartesian maps.
- The category of monoids in $\mathbf{Poly}^{\mathbf{cart}}$ is equivalent to...
- ...the category of (one-object, nonsymmetric, infinitary) operads.
  - Example: List has a monoid structure in $\mathbf{Poly}^{\mathbf{cart}}$.
  - It corresponds to the "associative operad"; one op'n of each arity.
- If you want the cat'y of finitary operads, use monoids over List.

Idea: $p(1) =$ "operations", and $p_i =$ "arity of operation i".

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f^\sharp_i$ is an iso for each $i \in p(1)$.

A special case of a result by Gambino and Kock says the following.

- Let $\mathbf{Poly}^{\mathbf{cart}}$ denote the category of polynomials and cartesian maps.
- The category of monoids in $\mathbf{Poly}^{\mathbf{cart}}$ is equivalent to...
- ...the category of (one-object, nonsymmetric, infinitary) operads.
  - Example: List has a monoid structure in $\mathbf{Poly}^{\mathbf{cart}}$.
  - It corresponds to the "associative operad"; one op'n of each arity.
- If you want the cat'y of finitary operads, use monoids over List.

Idea: $p(1) =$ "operations", and $p_i =$ "arity of operation i".

- The unit map $\eta\colon y \to p$ picks out an operation to serve as identity.

(5) Operads as cartesian monoids

Recall: $(f, f^\sharp)\colon p \to q$ is *cartesian* if $f_i^\sharp$ is an iso for each $i \in p(1)$.

A special case of a result by Gambino and Kock says the following.

- Let $\mathbf{Poly}^{\mathbf{cart}}$ denote the category of polynomials and cartesian maps.
- The category of monoids in $\mathbf{Poly}^{\mathbf{cart}}$ is equivalent to...
- ...the category of (one-object, nonsymmetric, infinitary) operads.
  - Example: List has a monoid structure in $\mathbf{Poly}^{\mathbf{cart}}$.
  - It corresponds to the "associative operad"; one op'n of each arity.
- If you want the cat'y of finitary operads, use monoids over List.

Idea: $p(1) =$ "operations", and $p_i =$ "arity of operation i".

- The unit map $\eta\colon y \to p$ picks out an operation to serve as identity.
- The map $\mu\colon p \circ p \to p$ gives the composition structure:
  - Given operation $i \in p(1)$ and, for each $d \in p_i$ an operation j_d...
  - ... get an operation to serve as $i \circ (j_1, \ldots, j_{p_i})$ with the right arity.

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

Answer: A comonoid (p, δ, ϵ) in $(\mathbf{Poly}, \circ, y)$ is precisely a *category*!

- Idea: the positions $p(1)$ are the objects of the category.

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

Answer: A comonoid (p, δ, ϵ) in $(\textbf{Poly}, \circ, y)$ is precisely a *category*!

- Idea: the positions $p(1)$ are the objects of the category.
- For each $i \in p(1)$, distinctions p_i are the outgoing morphisms $i \to$.

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

Answer: A comonoid (p, δ, ϵ) in $(\mathbf{Poly}, \circ, y)$ is precisely a *category*!

- Idea: the positions $p(1)$ are the objects of the category.
- For each $i \in p(1)$, distinctions p_i are the outgoing morphisms $i \to$.
- The counit $\epsilon\colon p \to y$ assigns an identity to each object.
- The comult'ation $\delta\colon p \to p \circ p$ assigns codomains and compositions.

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

Answer: A comonoid (p, δ, ϵ) in $(\mathbf{Poly}, \circ, y)$ is precisely a *category*!

- Idea: the positions $p(1)$ are the objects of the category.
- For each $i \in p(1)$, distinctions p_i are the outgoing morphisms $i \to$.
- The counit $\epsilon\colon p \to y$ assigns an identity to each object.
- The comult'ation $\delta\colon p \to p \circ p$ assigns codomains and compositions.
- The comonoid laws turn into category laws (unit, assoc.)

(6) Categories are comonoids

We asked above if there's a good way to think about comonoids in **Poly**.

Answer: A comonoid (p, δ, ϵ) in $(\textbf{Poly}, \circ, y)$ is precisely a *category*!

- Idea: the positions $p(1)$ are the objects of the category.
- For each $i \in p(1)$, distinctions p_i are the outgoing morphisms $i \to$.
- The counit $\epsilon\colon p \to y$ assigns an identity to each object.
- The comult'ation $\delta\colon p \to p \circ p$ assigns codomains and compositions.
- The comonoid laws turn into category laws (unit, assoc.)

This is due to Ahman and Uustalu. I learned it from Richard Garner.

- In fact, Garner offers a high-brow and a low-brow way to see this.
- I'll discuss each.

Another way to think about maps in Poly

To setup Garner's machine, recall one of the caty's equivalent to **Poly**.

Another way to think about maps in Poly

To setup Garner's machine, recall one of the caty's equivalent to **Poly**.

- Objects are typed sets $\tau\colon S \to \mathbf{Set}$,
- Morphisms are colax triangles:

$$\begin{tikzcd}
S \arrow[rr, "f"] \arrow[dr, "\tau"'] & {} \arrow[d, phantom, "\overset{f^\sharp}{\Leftarrow}"] & S' \arrow[dl, "\tau'"] \\
& \mathbf{Set} &
\end{tikzcd}$$

Another way to think about maps in Poly

To setup Garner's machine, recall one of the caty's equivalent to **Poly**.

- Objects are typed sets $\tau\colon S \to \mathbf{Set}$,
- Morphisms are colax triangles:

$$\begin{tikzcd}
S \arrow[rr, "f"] \arrow[dr, "\tau"', ""{name=L}] & & S' \arrow[dl, "\tau'", ""'{name=R}] \\
& \mathbf{Set} &
\arrow[Rightarrow, from=R, to=L, "f^\sharp"']
\end{tikzcd}$$

This is equivalent to the following:

- Objects: functions $\pi\colon E \to B$ between sets.
- Morphisms $\mathbf{Poly}(\pi, \pi') = \{(f, f^\sharp) \mid f\colon B \to B', f^\sharp\colon f^*E' \to E\}$.

$$\begin{tikzcd}
E \arrow[d, "\pi"'] & f^*E' \arrow[l, "f^\sharp"'] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E' \arrow[d, "\pi'"] \\
B \arrow[r, equal] & B \arrow[r, "f"'] & B'
\end{tikzcd}$$

Garner's bouquets of pullbacks

In his video (link later), Garner gives a calculus for maps $p \to p_1 \circ \cdots \circ p_n$.

- Say $p = E \to B$ and $p_i = E_i \to B_i$ for $i = 1, \ldots, n$.

Garner's bouquets of pullbacks

In his video (link later), Garner gives a calculus for maps $p \to p_1 \circ \cdots \circ p_n$.

- Say $p = E \to B$ and $p_i = E_i \to B_i$ for $i = 1, \ldots, n$.
- We can identify a map $p \to p_1 \circ \cdots \circ p_n$ with the $n{+}1$ blue arrows:

$$\begin{tikzcd}
 & & & & E_n \arrow[ddr, "p_n"] & \\
 & & \bullet \arrow[urr, dashed] \arrow[dll, blue, "f^\sharp"'] \arrow[ddr, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & & & \\
E \arrow[dddddd, "p"'] & & & & & B_n \\
 & & & \bullet \arrow[urr, blue, "f_n"] \arrow[rr, dashed] \arrow[d, dotted, no head] & & E_{n-1} \arrow[d, dotted, no head] \\
 & & & \vdots & & \vdots \\
 & & & \bullet \arrow[rr, blue, "f_3"] \arrow[drr, dashed] \arrow[ddl, dashed] \arrow[d, phantom, "\llcorner", very near start] & & B_3 \\
 & & & & & E_2 \arrow[ddl, "p_2"] \\
 & & \bullet \arrow[dll, dashed] \arrow[drr, blue, "f_2"] \arrow[ddr, dashed] \arrow[d, phantom, "\llcorner", very near start] & & & \\
B \arrow[ddr, blue, "f_1"] & & & & B_2 & \\
 & & & E_1 \arrow[dll, "p_1"] & & \\
 & B_1 & & & &
\end{tikzcd}$$

Garner's bouquets of pullbacks

In his video (link later), Garner gives a calculus for maps $p \to p_1 \circ \cdots \circ p_n$.

- Say $p = E \to B$ and $p_i = E_i \to B_i$ for $i = 1, \ldots, n$.
- We can identify a map $p \to p_1 \circ \cdots \circ p_n$ with the $n{+}1$ blue arrows:

$$\begin{tikzcd}
 & & & E_n \arrow[dr, "p_n"] & \\
 & & \bullet \arrow[ur, dashed] \arrow[dll, blue, "f^\sharp"'] \arrow[dr, dashed] \arrow[dr, phantom, very near start, "\lrcorner"] & & B_n \\
E \arrow[dddd, "p"'] & & & \bullet \arrow[ur, blue, "f_n"] \arrow[r, dashed] \arrow[d, dotted, no head] & E_{n-1} \arrow[d, dotted, no head] \\
 & & & \bullet \arrow[r, blue, "f_3"] \arrow[dr, dashed] \arrow[ddl, dashed] \arrow[d, phantom, very near start, "\llcorner"] & B_3 \\
 & & & & E_2 \arrow[ddl, "p_2"] \\
 & & \bullet \arrow[dll, dashed] \arrow[dr, blue, "f_2"] \arrow[ddr, dashed] \arrow[d, phantom, very near start, "\llcorner"] & & \\
B \arrow[ddr, blue, "f_1"] & & & B_2 & \\
 & & & E_1 \arrow[dll, "p_1"] & \\
 & B_1 & & &
\end{tikzcd}$$

To compose, e.g. with $p_i \to q_1 \circ q_m$, take a lot of pullbacks.

Painstaking calculation that comonoids = categories

One can perform a $15^{?}$-minute calculation to prove comonoids=categories.

$$\begin{tikzcd}
C_1 \arrow[dd, "dom"'] & C_0 \arrow[l, "id"'] \arrow[dr, dashed, "!"] \arrow[ddl, equal, dashed] \arrow[dd, phantom, "\vee", very near start] & \\
& & 1 \arrow[dl, equal] \\
C_0 \arrow[r, "!"] & 1 &
\end{tikzcd}$$

$$\begin{tikzcd}
& C_2 \arrow[dl, "comp"'] \arrow[rr, dashed] \arrow[dd, dashed] \arrow[ddrr, phantom, "\lrcorner", very near start] & & C_1 \arrow[dd, "dom"] \\
C_1 \arrow[dd, "dom"'] & & & \\
& C_1 \arrow[rr, "cod"] \arrow[dl, dashed] \arrow[dr, equal, dashed] \arrow[dd, phantom, "\vee", very near start] & & C_0 \\
C_0 \arrow[dr, equal, "(*)"] & & C_1 \arrow[dl, "dom"] & \\
& C_0 & &
\end{tikzcd}$$

Painstaking calculation that comonoids = categories

One can perform a $15^?$-minute calculation to prove comonoids=categories.

$$\begin{tikzcd}
C_1 \arrow[d, "dom"'] & C_0 \arrow[l, blue, "id"'] \arrow[dr, dashed, "!"] \arrow[dl, dashed, equal] \arrow[d, phantom, "\vee" near start] & \\
C_0 \arrow[dr, blue, "!"] & & 1 \arrow[dl, equal] \\
& 1 &
\end{tikzcd}
\qquad
\begin{tikzcd}
& C_2 \arrow[r, dashed] \arrow[dl, blue, "comp"'] \arrow[dd, dashed] \arrow[dr, phantom, "\lrcorner" very near start] & C_1 \arrow[dd, "dom"] \\
C_1 \arrow[dd, "dom"'] & & \\
& C_1 \arrow[r, blue, "cod"] \arrow[dl, dashed] \arrow[dr, dashed, equal] \arrow[d, phantom, "\vee" near start] & C_0 \\
C_0 \arrow[dr, blue, equal, "(*)"] & & C_1 \arrow[dl, "dom"] \\
& C_0 &
\end{tikzcd}$$

The comonoid laws enforce the necessary equations

- In particular, the fact that $(*)$ is identity is forced by unitality.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
  - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
  - Other way takes work. Ask after if you want a few more details.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
  - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
  - Other way takes work. Ask after if you want a few more details.
- Any $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits has a left adjoint.
- So R leads to a monoid $\mathbf{Set}/A \xrightarrow{L} \mathbf{Set}/A$ preserving all colimits.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
  - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
  - Other way takes work. Ask after if you want a few more details.
- Any $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits has a left adjoint.
- So R leads to a monoid $\mathbf{Set}/A \xrightarrow{L} \mathbf{Set}/A$ preserving all colimits.
  - Any such functor is determined by its image on elements of A.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
  - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
  - Other way takes work. Ask after if you want a few more details.
- Any $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits has a left adjoint.
- So R leads to a monoid $\mathbf{Set}/A \xrightarrow{L} \mathbf{Set}/A$ preserving all colimits.
  - Any such functor is determined by its image on elements of A.
  - For each $a \in A$ get a set $L(a)$ mapping to A.
  - We can identify the functor L with a span $A \leftarrow \bullet \rightarrow A$.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** → **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
    - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
    - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
    - Other way takes work. Ask after if you want a few more details.
- Any $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits has a left adjoint.
- So R leads to a monoid $\mathbf{Set}/A \xrightarrow{L} \mathbf{Set}/A$ preserving all colimits.
    - Any such functor is determined by its image on elements of A.
    - For each $a \in A$ get a set $L(a)$ mapping to A.
    - We can identify the functor L with a span $A \leftarrow \bullet \rightarrow A$.
- So monoids in colim-preserving $\mathbf{Set}/A \rightarrow \mathbf{Set}/A$ are monoids in **Span**.

High-brow explanation that comonoids = categories

I also learned the following high-brow explanation from Garner.

- **Poly** is the cat'y of functors **Set** $\to$ **Set** preserving connected limits.
- A comonoid in **Poly** can be identified with:
  - a set A and a comonad $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits.
  - One way's easy: send R to $\Delta_A \mathbin{⨟} R \mathbin{⨟} \Sigma_A$, which prsrvs conn limits.
  - Other way takes work. Ask after if you want a few more details.
- Any $\mathbf{Set}/A \xrightarrow{R} \mathbf{Set}/A$ preserving all limits has a left adjoint.
- So R leads to a monoid $\mathbf{Set}/A \xrightarrow{L} \mathbf{Set}/A$ preserving all colimits.
  - Any such functor is determined by its image on elements of A.
  - For each $a \in A$ get a set $L(a)$ mapping to A.
  - We can identify the functor L with a span $A \leftarrow \bullet \rightarrow A$.
- So monoids in colim-preserving $\mathbf{Set}/A \to \mathbf{Set}/A$ are monoids in **Span**.
- Monoids in **Span** are the same as small categories. *Tada..!*

Back to the coalgebra picture

What does "comonoids=categories" mean in the setting of coalgebras?

- We said earlier that a p-coalgebra S is just a map $Sy^S \to p$.
- And we said that what makes it work is that Sy^S is a comonoid.
- But comonoids in $(\mathbf{Poly}, \circ, y)$ are categories.
- So what is this comonoid Sy^S as a category?

Back to the coalgebra picture

What does "comonoids=categories" mean in the setting of coalgebras?

- We said earlier that a p-coalgebra S is just a map $Sy^S \to p$.
- And we said that what makes it work is that Sy^S is a comonoid.
- But comonoids in $(\mathbf{Poly}, \circ, y)$ are categories.
- So what is this comonoid Sy^S as a category?
- It's the contractible groupoid on S; objects=S, a unique map $s \to s'$.

Back to the coalgebra picture

What does "comonoids=categories" mean in the setting of coalgebras?

- We said earlier that a p-coalgebra S is just a map $Sy^S \to p$.
- And we said that what makes it work is that Sy^S is a comonoid.
- But comonoids in $(\mathbf{Poly}, \circ, y)$ are categories.
- So what is this comonoid Sy^S as a category?
- It's the contractible groupoid on S; objects$=S$, a unique map $s \to s'$.

A more general notion of dynamical system inhabiting interface $p \in \mathbf{Poly}$:

- A choice of comonoid (C, δ, ϵ) and a map $C \to p$.

Back to the coalgebra picture

What does "comonoids=categories" mean in the setting of coalgebras?

- We said earlier that a p-coalgebra S is just a map $Sy^S \to p$.
- And we said that what makes it work is that Sy^S is a comonoid.
- But comonoids in $(\mathbf{Poly}, \circ, y)$ are categories.
- So what is this comonoid Sy^S as a category?
- It's the contractible groupoid on S; objects$=S$, a unique map $s \to s'$.

A more general notion of dynamical system inhabiting interface $p \in \mathbf{Poly}$:

- A choice of comonoid (C, δ, ϵ) and a map $C \to p$.
- In general C keeps track of the internal path history through time.

Back to the coalgebra picture

What does "comonoids=categories" mean in the setting of coalgebras?

- We said earlier that a p-coalgebra S is just a map $Sy^S \to p$.
- And we said that what makes it work is that Sy^S is a comonoid.
- But comonoids in $(\mathbf{Poly}, \circ, y)$ are categories.
- So what is this comonoid Sy^S as a category?
- It's the contractible groupoid on S; objects$=S$, a unique map $s \to s'$.

A more general notion of dynamical system inhabiting interface $p \in \mathbf{Poly}$:

- A choice of comonoid (C, δ, ϵ) and a map $C \to p$.
- In general C keeps track of the internal path history through time.
  - That is, a history of interaction is recorded as a morphism in C.
  - When $C = Sy^S$, a history is just a start state and an end state.

$\mathbf{Cat}^{\sharp} := \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

$\mathbf{Cat}^\sharp \coloneqq \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
    - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
    - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp\colon D(fc, -) \longrightarrow C(c, -)$...
    - ... such that $f^\sharp$ preserves identity and composition.

$\mathbf{Cat}^\sharp$:= Comon(Poly): categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp\colon D(fc, -) \longrightarrow C(c, -)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, "monoid of naturals".

$\mathbf{Cat}^\sharp := \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp\colon D(fc, -) \longrightarrow C(c, -)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, “monoid of naturals”.
  - A cofunctor $C \to N$ is just a polynomial map $C \to y$.
  - It's a trajectory: a choice of morphism $f_c\colon c \to ?$ out of each c.

$\mathbf{Cat}^\sharp \coloneqq \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f^\sharp_c\colon D(fc, -) \longrightarrow C(c, -)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, "monoid of naturals".
  - A cofunctor $C \to N$ is just a polynomial map $C \to y$.
  - It's a trajectory: a choice of morphism $f_c\colon c \to\ ?$ out of each c.

$$\begin{tikzcd}
\mathbf{Mon}^{\mathrm{op}} \arrow[r, shift left=2, "y^{-}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Cat}^\sharp \arrow[l, shift left=2, "\mathbf{Comon}(-{,}y^{\mathbb{N}})"]
\end{tikzcd}$$

$\mathbf{Cat}^\sharp := \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp\colon D(fc,-) \longrightarrow C(c,-)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, "monoid of naturals".
  - A cofunctor $C \to N$ is just a polynomial map $C \to y$.
  - It's a trajectory: a choice of morphism $f_c\colon c \to ?$ out of each c.

$$\begin{tikzcd}
\mathbf{Mon}^{\mathrm{op}} \arrow[r, shift left=2, "y^{-}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Cat}^\sharp \arrow[l, shift left=2, "\mathbf{Comon}(-{,}y^{\mathbb{N}})"]
\end{tikzcd}$$

So we can think of $\mathbf{Cat}^\sharp$ as the category of categories and cofunctors.

$\mathbf{Cat}^\sharp := \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f \colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp \colon D(fc, -) \longrightarrow C(c, -)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, "monoid of naturals".
  - A cofunctor $C \to N$ is just a polynomial map $C \to y$.
  - It's a trajectory: a choice of morphism $f_c \colon c \to ?$ out of each c.

$$\begin{tikzcd}
\mathbf{Mon}^{\mathrm{op}} \arrow[r, shift left=2, "y^{-}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Cat}^\sharp \arrow[l, shift left=2, "\mathbf{Comon}(-{,}y^{\mathbb{N}})"]
\end{tikzcd}$$

So we can think of $\mathbf{Cat}^\sharp$ as the category of categories and cofunctors.

- It has two mon'l structures, $+, \otimes$; remember the duoidal structures?

$\mathbf{Cat}^\sharp := \mathbf{Comon}(\mathbf{Poly})$: categories and cofunctors

Comonoids are small categories; comonoid morphisms are *cofunctors*.

- A cofunctor $C \to D$ consists of:
  - A function $f\colon \operatorname{Ob} C \to \operatorname{Ob} D$ and...
  - ... for all $c \in \operatorname{Ob}(C)$ a function $f_c^\sharp\colon D(fc, -) \longrightarrow C(c, -)$...
  - ... such that $f^\sharp$ preserves identity and composition.
- Example: let $N = y^{\mathbb{N}} = \mathbf{Cofree}(y)$, "monoid of naturals".
  - A cofunctor $C \to N$ is just a polynomial map $C \to y$.
  - It's a trajectory: a choice of morphism $f_c\colon c \to\ ?$ out of each c.

$$\begin{tikzcd} \mathbf{Mon}^{\mathrm{op}} \arrow[r, shift left=2, "y^{-}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Cat}^\sharp \arrow[l, shift left=2, "\mathbf{Comon}(-{,}y^{\mathbb{N}})"] \end{tikzcd}$$

So we can think of $\mathbf{Cat}^\sharp$ as the category of categories and cofunctors.

- It has two mon'l structures, $+, \otimes$; remember the duoidal structures?
- The sum $C + D$ returns the coproduct of categories.
- The Dirichlet product $C \otimes D$ returns the product of categories.
- Note that $+$ is not a coproduct and $\otimes$ is not a product in $\mathbf{Cat}^\sharp$.

(7) Bimodules are parametric right adjoints

The last of the wonders of $(\mathbf{Poly}, \circ, y)$ is again due to Garner.

(7) Bimodules are parametric right adjoints

The last of the wonders of $(\textbf{Poly}, \circ, y)$ is again due to Garner.

- Let C, D be comonoids (categories); a (C, D)-bimodule consists of:
- a polynomial M and maps $C \circ M \xleftarrow{f} M \xrightarrow{g} M \circ D$ such that

(7) Bimodules are parametric right adjoints

The last of the wonders of $(\mathbf{Poly}, \circ, y)$ is again due to Garner.

- Let C, D be comonoids (categories); a (C, D)-bimodule consists of:
- a polynomial M and maps $C \circ M \xleftarrow{f} M \xrightarrow{g} M \circ D$ such that
  - $f \mathbin{;} (\epsilon_C \circ M) = M$ and similar for g;
  - $f \mathbin{;} (\delta_C \circ M) = f \mathbin{;} (C \circ f)$ and similarly for g; and
  - $f \mathbin{;} (C \circ g) = g \mathbin{;} (f \circ D)$ as maps $M \to C \circ M \circ D$.

(7) Bimodules are parametric right adjoints

The last of the wonders of $(\mathbf{Poly}, \circ, y)$ is again due to Garner.

- Let C, D be comonoids (categories); a (C, D)-bimodule consists of:
- a polynomial M and maps $C \circ M \xleftarrow{f} M \xrightarrow{g} M \circ D$ such that
    - $f \mathbin{⨾} (\epsilon_C \circ M) = M$ and similar for g;
    - $f \mathbin{⨾} (\delta_C \circ M) = f \mathbin{⨾} (C \circ f)$ and similarly for g; and
    - $f \mathbin{⨾} (C \circ g) = g \mathbin{⨾} (f \circ D)$ as maps $M \to C \circ M \circ D$.

These are exactly parametric right adjoints (pra's) $F \colon \mathbf{Set}^D \to \mathbf{Set}^C$!

(7) Bimodules are parametric right adjoints

The last of the wonders of $(\mathbf{Poly}, \circ, y)$ is again due to Garner.

- Let C, D be comonoids (categories); a (C, D)-bimodule consists of:
- a polynomial M and maps $C \circ M \xleftarrow{f} M \xrightarrow{g} M \circ D$ such that
    - $f \mathbin{⨟} (\epsilon_C \circ M) = M$ and similar for g;
    - $f \mathbin{⨟} (\delta_C \circ M) = f \mathbin{⨟} (C \circ f)$ and similarly for g; and
    - $f \mathbin{⨟} (C \circ g) = g \mathbin{⨟} (f \circ D)$ as maps $M \to C \circ M \circ D$.

These are exactly parametric right adjoints (pra's) $F \colon \mathbf{Set}^D \to \mathbf{Set}^C$!

- Let $1 \in \mathbf{Set}^D$ be the terminal object; then we can lift F to...
- ...$F' \colon \mathbf{Set}^D \to \mathbf{Set}^C/(F1)$. Say F is a *pra* if F' is a right adjoint.

Relation to databases

This all fits surprisingly well with the “databases as categories” story.

Relation to databases

This all fits surprisingly well with the "databases as categories" story.

- A *database schema* is a network of interconnected tables.
  - There's a collection of tables, each with a collection of columns.
  - Each column points to a "foreign" table, a codomain.
  - There are *integrity constraints*.
  - Get a category C: objects=tables, maps out of c = columns of c.
  - So schema=category. The data itself is a functor $I\colon C \to \mathbf{Set}$.

Relation to databases

This all fits surprisingly well with the "databases as categories" story.

- A *database schema* is a network of interconnected tables.
  - There's a collection of tables, each with a collection of columns.
  - Each column points to a "foreign" table, a codomain.
  - There are *integrity constraints*.
  - Get a category C: objects=tables, maps out of c = columns of c.
  - So schema=category. The data itself is a functor $I: C \to \mathbf{Set}$.
- Polynomial comonoids C fits this "objects + maps-out" articulation.

Relation to databases

This all fits surprisingly well with the “databases as categories” story.

- A *database schema* is a network of interconnected tables.
    - There’s a collection of tables, each with a collection of columns.
    - Each column points to a “foreign” table, a codomain.
    - There are *integrity constraints*.
    - Get a category C: objects=tables, maps out of c = columns of c.
    - So schema=category. The data itself is a functor $I\colon C \to \textbf{Set}$.
- Polynomial comonoids C fits this “objects + maps-out” articulation.
- Coalgebras on C are set-valued functors $I\colon C \to \textbf{Set}$.
    - Let $S := \sum_{c \in C} I(c)$; then I can be encoded as coalg, $S \to C(S)$.
    - One can think of $S \to C \circ S$ as a constant (C, y)-bimodule.

Relation to databases

This all fits surprisingly well with the "databases as categories" story.

- A *database schema* is a network of interconnected tables.
    - There's a collection of tables, each with a collection of columns.
    - Each column points to a "foreign" table, a codomain.
    - There are *integrity constraints*.
    - Get a category C: objects=tables, maps out of c = columns of c.
    - So schema=category. The data itself is a functor $I\colon C \to \mathbf{Set}$.
- Polynomial comonoids C fits this "objects + maps-out" articulation.
- Coalgebras on C are set-valued functors $I\colon C \to \mathbf{Set}$.
    - Let $S := \sum_{c\in C} I(c)$; then I can be encoded as coalg, $S \to C(S)$.
    - One can think of $S \to C \circ S$ as a constant $(C, \mathcal{y})$-bimodule.
- Bimodules are data-migration functors.
    - A (D, C)-bimodule is a parametric right adjoint $\mathbf{Set}^C \to \mathbf{Set}^D$.
    - These functors move database instances between schemas; useful!

Outline

1. Introduction
2. Categorical virtues of Poly
3. Composition product in dynamical systems
4. Theoretical wonders of the $\circ$ monoidal structure
5. **Conclusion**

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.
- Its $(\circ, y)$ monoidal structure has lots of surprises.
  - E.g. operads, categories, copresheaves, parametric right adjoints.

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.
- Its $(\circ, y)$ monoidal structure has lots of surprises.
    - E.g. operads, categories, copresheaves, parametric right adjoints.
    - Strategies, speeding up dynamics, generalizing coalgebras.

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.
- Its $(\circ, y)$ monoidal structure has lots of surprises.
  - E.g. operads, categories, copresheaves, parametric right adjoints.
  - Strategies, speeding up dynamics, generalizing coalgebras.
  - **Cat**$^\sharp$, categories and cofunctors, deserves more exploration.

Poly is really nice for thinking about dynamical systems and databases.

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.
- Its $(\circ, y)$ monoidal structure has lots of surprises.
  - E.g. operads, categories, copresheaves, parametric right adjoints.
  - Strategies, speeding up dynamics, generalizing coalgebras.
  - $\mathbf{Cat}^\sharp$, categories and cofunctors, deserves more exploration.

Poly is really nice for thinking about dynamical systems and databases.

- It encodes mode-dependent wiring diagrams and dynamical systems.
- It has direct application to databases and data migration.
- I wonder how these two applications might come together....

Summary

Poly is a beautiful category.

- There are many equivalent ways to formulate it.
- It has four interacting monoidal structures, two closures, etc.
- Its $(\circ, y)$ monoidal structure has lots of surprises.
    - E.g. operads, categories, copresheaves, parametric right adjoints.
    - Strategies, speeding up dynamics, generalizing coalgebras.
    - **Cat**$^\sharp$, categories and cofunctors, deserves more exploration.

Poly is really nice for thinking about dynamical systems and databases.

- It encodes mode-dependent wiring diagrams and dynamical systems.
- It has direct application to databases and data migration.
- I wonder how these two applications might come together....

Thanks; comments and questions welcome!

References

- Abbot, *Categories of containers*.
- Ahman-Uustalu, *Directed Containers as Categories*.
- Gambino-Kock, *Polynomial functors and polynomial monads*.
- Garner, HoTTest seminar 2019/12/12: `youtu.be/0o9HzQ3zAcE`.
- Myers-Spivak, *Dirichlet Functors are Contravariant Polynomial Functors*.
- Shulman, *Framed bicategories and monoidal fibrations*.
- Spivak, *Poly: An abundant categorical setting for mode-dependent dynamics*.
- Spivak, MIT CT seminar 2020/03/05: `youtu.be/U-W7GT0BUTU`.
- Spivak-Tan, *Nesting of dynamic systems and mode-dependent networks*.

Supplementary material

- Composing bimodules
- Example of cofree comonoids and relation to data
- Some details of Garner's equivalence proof

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$\begin{tikzcd} Eq \arrow[r] & M \circ N \arrow[r, shift left] \arrow[r, shift right] & M \circ D \circ N \end{tikzcd} \tag{1}$$

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$Eq \longrightarrow M \circ N \rightrightarrows M \circ D \circ N \tag{1}$$

- Claim: the following is an equalizer too:

$$C \circ (Eq) \circ E \longrightarrow C \circ (M \circ N) \circ E \rightrightarrows C \circ (M \circ D \circ N) \circ E \tag{2}$$

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$Eq \longrightarrow M \circ N \rightrightarrows M \circ D \circ N \tag{1}$$

- Claim: the following is an equalizer too:

$$C \circ (Eq) \circ E \longrightarrow C \circ (M \circ N) \circ E \rightrightarrows C \circ (M \circ D \circ N) \circ E \tag{2}$$

  - Equalizers in **Poly** are pointwise, so can postcompose (1) with E.

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$Eq \longrightarrow M \circ N \rightrightarrows M \circ D \circ N \tag{1}$$

- Claim: the following is an equalizer too:

$$C \circ (Eq) \circ E \longrightarrow C \circ (M \circ N) \circ E \rightrightarrows C \circ (M \circ D \circ N) \circ E \tag{2}$$

  - Equalizers in **Poly** are pointwise, so can postcompose (1) with E.
  - Poly's preserve connected limits, so can precompose (1) with C.

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$\begin{tikzcd} Eq \arrow[r] & M \circ N \arrow[r, shift left] \arrow[r, shift right] & M \circ D \circ N \end{tikzcd} \tag{1}$$

- Claim: the following is an equalizer too:

$$\begin{tikzcd} C \circ (Eq) \circ E \arrow[r] & C \circ (M \circ N) \circ E \arrow[r, shift left] \arrow[r, shift right] & C \circ (M \circ D \circ N) \circ E \end{tikzcd} \tag{2}$$

  - Equalizers in **Poly** are pointwise, so can postcompose (1) with E.
  - Poly's preserve connected limits, so can precompose (1) with C.
- Using the maps $C \circ M \leftarrow M$ and $M \rightarrow N \circ E$ gives (1) →(2).

Composing bimodules

To compose bimodules $C \circ M \leftarrow M \rightarrow M \circ D$ and $D \circ N \leftarrow N \rightarrow N \circ E$.

- The carrier of the composite will be the following equalizer:

$$\begin{tikzcd} Eq \arrow[r] & M \circ N \arrow[r, shift left] \arrow[r, shift right] & M \circ D \circ N \end{tikzcd} \tag{1}$$

- Claim: the following is an equalizer too:

$$\begin{tikzcd} C \circ (Eq) \circ E \arrow[r] & C \circ (M \circ N) \circ E \arrow[r, shift left] \arrow[r, shift right] & C \circ (M \circ D \circ N) \circ E \end{tikzcd} \tag{2}$$

  - Equalizers in **Poly** are pointwise, so can postcompose (1) with E.
  - Poly's preserve connected limits, so can precompose (1) with C.
- Using the maps $C \circ M \leftarrow M$ and $M \rightarrow N \circ E$ gives (1) →(2).
- Thus we get a new bimodule $C \circ Eq \leftarrow Eq \rightarrow Eq \circ E$, as desired.

Example of cofree comonoids and relation to data

Example of the cofree comonoid construction, namely on $p := By^A$:

Example of cofree comonoids and relation to data

Example of the cofree comonoid construction, namely on $p \coloneqq By^A$:

- The carrier of $C_p \coloneqq \textbf{Cofree}(p)$ is $(By)^{\mathrm{List}(A)}$.
- So as a category, C_p has an object for every function $r\colon \mathrm{List}(A) \to B$.

Example of cofree comonoids and relation to data

Example of the cofree comonoid construction, namely on $p \coloneqq By^A$:

- The carrier of $C_p \coloneqq \mathbf{Cofree}(p)$ is $(By)^{\mathrm{List}(A)}$.
- So as a category, C_p has an object for every function $r\colon \mathrm{List}(A) \to B$.
- $\mathrm{Hom}_{C_p}(r_1, r_2) = \{\ell \in \mathrm{List}(A) \mid \forall \ell' \in \mathrm{List}(A), r_1(\ell +\!\!+\, \ell') = r_2(\ell')\}$.

Example of cofree comonoids and relation to data

Example of the cofree comonoid construction, namely on $p \coloneqq By^A$:

- The carrier of $C_p \coloneqq \textbf{Cofree}(p)$ is $(By)^{\mathrm{List}(A)}$.
- So as a category, C_p has an object for every function $r \colon \mathrm{List}(A) \to B$.
- $\mathrm{Hom}_{C_p}(r_1, r_2) = \{\ell \in \mathrm{List}(A) \mid \forall \ell' \in \mathrm{List}(A), r_1(\ell +\!\!+ \ell') = r_2(\ell')\}$.

A dynamical system $(f, f^\sharp) \colon Sy^S \to p$ turns into a database instance on C_p.

Example of cofree comonoids and relation to data

Example of the cofree comonoid construction, namely on $p \coloneqq By^A$:

- The carrier of $C_p \coloneqq \textbf{Cofree}(p)$ is $(By)^{\mathrm{List}(A)}$.
- So as a category, C_p has an object for every function $r\colon \mathrm{List}(A) \to B$.
- $\mathrm{Hom}_{C_p}(r_1, r_2) = \{\ell \in \mathrm{List}(A) \mid \forall \ell' \in \mathrm{List}(A), r_1(\ell \mathbin{+\!\!+} \ell') = r_2(\ell')\}$.

A dynamical system $(f, f^\sharp)\colon Sy^S \to p$ turns into a database instance on C_p.

- Put each $s \in S$ into the table $r\colon \mathrm{List}(A) \to B$ that “acts like s”....
- ...that is, for which $r(\ell) = f(f^\sharp(\cdots f^\sharp(s, \ell_1), \cdots, \ell_n))$ for all ℓ.
- A table is a possible process. Its rows are the states that accomplish it.

Some details of Garner's equivalence proof

Claim:

$$\Sigma_{A\in\mathbf{Set}}\mathbf{Comon}(\mathbf{Set}/A \xrightarrow{lims^{\checkmark}} \mathbf{Set}/A) \cong \mathbf{Comon}(\mathbf{Set} \xrightarrow{conn-lims^{\checkmark}} \mathbf{Set})$$

Some details of Garner's equivalence proof

Claim:

$$\Sigma_{A\in\mathbf{Set}}\mathbf{Comon}(\mathbf{Set}/A \xrightarrow{\mathit{lims}^{\checkmark}} \mathbf{Set}/A) \cong \mathbf{Comon}(\mathbf{Set} \xrightarrow{\mathit{conn}-\mathit{lims}^{\checkmark}} \mathbf{Set})$$

Justification:

$(\rightarrow)$: send $R\colon \mathbf{Set}/A \xrightarrow{\mathit{lims}^{\checkmark}} \mathbf{Set}/A$ to $\Delta_A \mathbin{⨾} R \mathbin{⨾} \Sigma_A\colon \mathbf{Set} \xrightarrow{\mathit{conn}-\mathit{lims}^{\checkmark}} \mathbf{Set}$.

$$\begin{tikzcd}
\mathbf{Set} \arrow[r] \arrow[d, "\Delta_A"'] & \mathbf{Set} \\
\mathbf{Set}/A \arrow[r] & \mathbf{Set}/A \arrow[u, "\Sigma_A"']
\end{tikzcd}$$

Some details of Garner's equivalence proof

Claim:

$$\Sigma_{A\in\mathbf{Set}}\mathbf{Comon}(\mathbf{Set}/A \xrightarrow{\mathit{lims}^{\checkmark}} \mathbf{Set}/A) \cong \mathbf{Comon}(\mathbf{Set} \xrightarrow{\mathit{conn-lims}^{\checkmark}} \mathbf{Set})$$

Justification:

($\rightarrow$): send $R\colon \mathbf{Set}/A \xrightarrow{\mathit{lims}^{\checkmark}} \mathbf{Set}/A$ to $\Delta_A \fatsemi R \fatsemi \Sigma_A\colon \mathbf{Set} \xrightarrow{\mathit{conn-lims}^{\checkmark}} \mathbf{Set}$.

$$\begin{tikzcd}
\mathbf{Set} \arrow[r] \arrow[d, "\Delta_A"'] & \mathbf{Set} \\
\mathbf{Set}/A \arrow[r] & \mathbf{Set}/A \arrow[u, "\Sigma_A"']
\end{tikzcd}$$

($\leftarrow$): Given $Q\colon \mathbf{Set} \xrightarrow{\mathit{conn-lims}^{\checkmark}} \mathbf{Set}$, let $A = Q1$, send $\alpha\colon X \rightarrow Q1$ to α':

$$\begin{tikzcd}
\bullet \arrow[r] \arrow[d, "\alpha'"'] \arrow[dr, phantom, "\lrcorner", very near start] & QX \arrow[d, "Q\alpha"] \\
Q1 \arrow[r, "\delta_1"'] & QQ1
\end{tikzcd}$$

One can check that these two constructions are mutually inverse.