

Poly: An abundant categorical setting for mode-dependent dynamics

David I. Spivak

Abstract

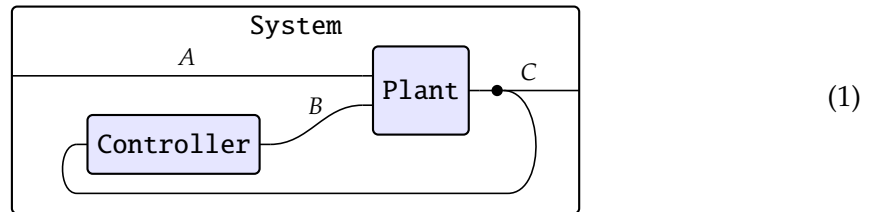
Dynamical systems—by which we mean machines that take time-varying input, change their state, and produce output—can be wired together to form more complex systems. Previous work has shown how to allow collections of machines to reconfigure their wiring diagram dynamically, based on their collective state. This notion was called “mode dependence”, and while the framework was compositional (forming an operad of re-wiring diagrams and algebra of mode-dependent dynamical systems on it), the formulation itself was more “creative” than it was natural.

In this paper we show that the theory of mode-dependent dynamical systems can be more naturally recast within the category *Poly* of polynomial functors. This category is almost superlatively abundant in its structure: for example, it has *four* interacting monoidal structures $(+, \times, \otimes, \circ)$, two of which (\times, \otimes) are monoidal closed, and the comonoids for \circ are precisely categories in the usual sense. We discuss how the various structures in *Poly* show up in the theory of dynamical systems. We also show that the usual coalgebraic formalism for dynamical systems takes place within *Poly*. Indeed one can see coalgebras as special dynamical systems—ones that do not record their history—formally analogous to contractible groupoids as special categories.

1 Introduction

We propose the category *Poly* of polynomial functors on *Set* as a setting in which to model very general sorts of dynamics and interaction. Let’s back up and say what exactly it is that we’re generalizing.

A wiring diagram can be used to specify a fixed communication pattern between systems:



Shown here, the plant—say a power plant or a car—is a dynamical system that receives input of type A from the outside world and input of type B from the controller, and it produces output of type C ; this in turn is fed both to the outside world and to the controller. Given these fixed sets A, B, C , we will see shortly that the two interior boxes

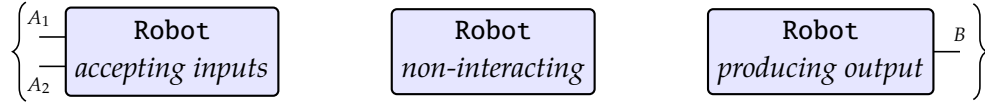
and one exterior box shown in (1) can be faithfully represented by polynomials in one variable y , as follows:

$$\text{Plant} = Cy^{AB} \quad \text{Controller} = By^C \quad \text{System} = Cy^A. \quad (2)$$

Observe that in each case the output type is the coefficient on y , and the input type is the exponent on y . In Section 3.3 we will see that the wiring diagram (1) itself, as well as the interacting dynamics, can be represented by morphisms involving these polynomials.

1.1 Introduction to mode-dependence

Notice that the polynomials in (2) are monomials; it is this we want to generalize. By using more general polynomials such as $\text{Robot} = y^{A_1 A_2} + y + By$, we can create a system for which the input-output types are not fixed:



What we discuss in this paper are dynamical systems whose interfaces change in time, and similarly where the wiring diagram connecting the systems changes in time. These changes will be based on the internal states of the systems—say robots—involved.

The real world is filled with instances of systems with time-varying input-output patterns. The network topology—the way that the system wires up—changes based on both internal and environmental contexts. Consider the following situations:

1. When too much force is applied to a material, bonds can break;



2. A company may change its supplier at any time;



3. When someone assembles a machine, their own outputs dictate the connection pattern of the machine's components.



We will discuss (3) and (4) further in Example 3.5. In each of the above cases the wiring diagram—the connection pattern—changes based on the states (position, decision-making, environmental context, etc.) of some or all the systems involved. In [ST17] this was called *mode-dependence*; the goal of that article was to create an operadic framework in which mode-dependent dynamics and communication could be specified compositionally. While successful, the presentation was fairly ad hoc. The purpose of the present

paper is to explain that the category \mathbf{Poly} provides an abundant setting in which to work quite naturally with mode-dependent dynamics.

When we say that \mathbf{Poly} is abundant, we mean that it is exceptionally rich in structure, and that structure is highly relevant to dynamical systems. Here are some of the features of this category:

1. \mathbf{Poly} has coproducts and products, $+$, \times , the usual sum and product of polynomials.
2. \mathbf{Poly} has two additional monoidal structures: \otimes and \circ .
3. \mathbf{Poly} has two monoidal closed structures: for \times (cartesian closure) and \otimes .
4. \mathbf{Poly} has a duoidal structure: $(\circ) \otimes (\circ) \rightarrow (\otimes) \circ (\otimes)$.
5. \mathbf{Poly} has all small limits and is extensive.
6. \mathbf{Poly} has two orthogonal factorization systems (epi/mono and vertical/cartesian).
7. \mathbf{Poly} admits a monoidal bifibration $\mathbf{Poly} \rightarrow \mathbf{Set}$ with $\otimes \mapsto \times$.
8. \mathbf{Poly} admits an adjoint quadruple with \mathbf{Set} and an adjoint pair with \mathbf{Set}^{op} .
9. Comonoids in (\mathbf{Poly}, \circ) are precisely categories in the usual sense.

In Section 2 we will introduce \mathbf{Poly} and many of its interesting features. In Section 3, we will discuss how these features relate to dynamical systems.

1.2 Acknowledgments

We thank Richard Garner and David Jaz Myers for helpful and interesting conversations. We also appreciate support from Honeywell and AFOSR grants FA9550-17-1-0058 and FA9550-19-1-0113.

2 Introduction to \mathbf{Poly}

2.1 Polynomial functors

Notation 2.1. We usually denote sets with upper-case letters A, B , etc.; the exception is ordinals: we denote the n th ordinal by $n = \{1, \dots, n\}$. We denote functions between sets—including elements of sets—using upright letters $f: A \rightarrow B$ and $a \in A$.

All polynomials discussed here have a single variable, always y ; in particular y itself is a polynomial. Coefficients and exponents of polynomials are arbitrary sets, e.g. $\mathbb{N}y^{\mathbb{R}} + 3$ is a polynomial. Every set A will also be a polynomial, namely a constant. We denote generic polynomials with lower-case letters p, q , etc.

Recall that a *representable functor* $\mathbf{Set} \rightarrow \mathbf{Set}$ is one of the form $\mathbf{Set}(A, -)$ for a set A . We denote this functor by $y^A: \mathbf{Set} \rightarrow \mathbf{Set}$ and say it is *represented by* $A \in \mathbf{Set}$. For example y^3 is represented by 3 and $y^3(2) \cong 8$. As A varies we obtain the contravariant Yoneda embedding.

Classically, a polynomial p in one variable with set coefficients is a function $p(y) = A_n y^n + \dots + A_1 y^1 + A_0 y^0$ with each $A_i \in \mathbb{N}$. In category theory this is often generalized to allow for infinitely many terms and infinite exponents; e.g. we consider the following to be a polynomial

$$p(y) = \sum_{i \in I} y^{A_i} \quad (5)$$

for arbitrary small sets I and A . We can think of such a p as a functor $\mathbf{Set} \rightarrow \mathbf{Set}$; it sends a set $X \in \mathbf{Ob}(\mathbf{Set})$ to the coproduct, over $i \in I$, of the set X^{A_i} of functions $A_i \rightarrow X$, or equivalently the A_i -fold product of X with itself. The result is covariantly functorial in X .

Considered this way, p is called a *polynomial functor*; polynomial functors sit inside of the category of all functors $\mathbf{Set} \rightarrow \mathbf{Set}$ as a full subcategory, namely the one spanned by coproducts of representables.

Definition 2.2. The category **Poly** has polynomial functors $p(y)$ as in (5) as objects and natural transformations between them as morphisms.

In **Poly**, products distribute over coproducts

$$\left(\sum_i p_i \right) \times q \cong \sum_i (p_i \times q).$$

In fact, **Poly** can be characterized as the free category that has both coproducts and also products that distribute over coproducts. **Poly** is also equivalent to the Grothendieck construction of the canonical functor $\mathbf{Set}^{\mathbf{op}} \rightarrow \mathbf{Cat}$ sending each object to the corresponding slice category $A \mapsto \mathbf{Set}/A$ and sending $f: B \rightarrow A$ to pullback along f .

Notation 2.3. We denote the product of polynomials by juxtaposition or sometimes \cdot , i.e.

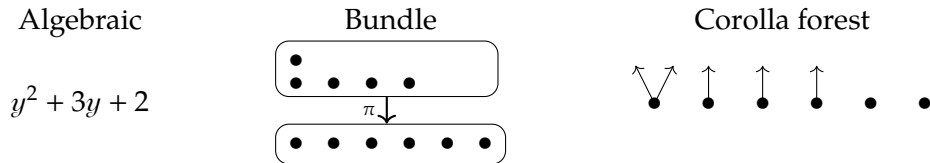
$$pq := p \times q = p \cdot q.$$

For any set A we denote the A -fold repeated product of p by $p^A := \prod_{a \in A} p$; in particular $p^1 \cong p$ and $p^0 \cong 1$. The representable y^n is indeed the n -fold repeated product of y .

For any polynomial p , the set $p(1)$ has particular importance; it can be identified with the set of representable summands (pure-power terms) y^k in p . For example if $p = y^2 + 3y + 2$ then $p(1) = 6$ corresponding to the six representable summands in $p = y^2 + y^1 + y^1 + y^1 + y^0 + y^0$. We will denote the representing object for the i th representable summand of p by p_i , i.e.

$$p = \sum_{i \in p(1)} y^{p_i}.$$

There are many ways to think about polynomials, and one becomes more versatile by being able to use different representations for different purposes. So far we have been writing polynomials in the typical algebraic style, but one can also represent them as bundles, as forests of corollas, or as dependent types.



Given a bundle $\pi: E \rightarrow B$, and element $b \in B$, we denote the fiber $\pi^{-1}(b)$ by E_b . We will refer to elements of B as *positions* and elements of E_b as the *directions* in position b . From the algebraic viewpoint, a position is a ‘pure-power’, or representable summand, and the associated direction-type is its ‘exponent’ or representing object; from the tree viewpoint, a position is a *root* and the associated directions are its *leaves*.

Polynomials can be implemented in a dependently typed programming language, such as Idris. Here is a specification of the type for polynomials:

```

record Poly where
  constructor MkPoly          -- To construct a poly, define:
  position  : Type            -- the "positions" (as a type), and
  direction : position -> Type -- the "directions" in each position.

```

For example, the expression `MkPoly Integer (\i => Double)` means that the type of positions is \mathbb{Z} and for each position the type of directions is the type `double` of double-precision floating point numbers; thinking of it as the reals, this denotes the monomial $\mathbb{Z}y^{\mathbb{R}}$. We will only discuss Idris once more in this document, though almost everything we discuss has been implemented; please write to the author for more information.

2.2 Morphisms of polynomials, concretely

As mentioned, the morphisms between polynomials are the natural transformations. As easy as this is to state—and as much as it gives us confidence in the reasonableness of the definition—it can be useful to have a more hands-on understanding of the morphisms.

By the Yoneda lemma, a morphism $y^A \rightarrow y^B$ can be identified with a function $B \rightarrow A$. One can prove that $+$ is the coproduct in `Poly` and \times is the product. Thus $y^2 + 3y + 2$ is a product of $y + 1$ and $y + 2$, and it is a coproduct of $y^2 + 1$ and $3y + 1$. This also holds for infinite sums and products: the usual algebraic operations coincide with the categorical operations. From this, and the fact that coproducts of functors $\mathbf{Set} \rightarrow \mathbf{Set}$ are taken pointwise, we obtain the following formula for the set of morphisms $p \rightarrow q$:

$$\text{Poly}(p, q) \cong \prod_{i \in p(1)} \sum_{j \in q(1)} p_i^{q_j}.$$

For each representable summand of p —i.e. position of p —choose a representable summand of q and give a function from the representing object (exponent) in q back to the representing object (exponent) in p . Thus for example $\text{Poly}(y^2 + 3y + 2, y^5 + 1) \cong (2^5 + 1)(1^5 + 1)(1^5 + 1)(1^5 + 1)(0^5 + 1)(0^5 + 1)$.

In terms of bundles, a morphism $E \rightarrow B$ to $E' \rightarrow B'$ consists of a pair $(f, f^\#)$ as shown:

$$\begin{array}{ccccc} E & \xleftarrow{f^\#} & B \times_{B'} E' & \longrightarrow & E' \\ \downarrow & & \downarrow & \lrcorner & \downarrow \\ B & \xlongequal{\quad} & B & \xrightarrow{f} & B' \end{array}$$

This will be the most convenient way to write morphisms of polynomials; we further denote by $f_i^\#$ the map on fibers $E'(f(p)) \rightarrow E(p)$. We refer to f as the *on-positions* function and $f^\#$ as the *on-directions* function. This way of thinking about morphisms of polynomials extends readily to Idris:

```

record Lens (dom : Poly) (cod : Poly) where
  constructor MkLens
  onPos : position dom -> position cod
  onDir : (i : position dom) -> direction cod j -> direction dom i
  where j = onPos i

```

The reason for the name *lens* comes from the following.

Example 2.4 (Bimorphic lenses). In [Hed18], Hedges defines the category of bimorphic lenses to have objects given by pairs of sets (A, B) and morphisms (called *lenses*) from (A, B) to (A', B') defined by a pair of maps $A \rightarrow A'$ and $A \times B' \rightarrow B$. It is straightforward to check that Hedges' category of bimorphic lenses is equivalent to the full subcategory of Poly spanned by the monomials By^A .

Monomials By^A will play a special role in the theory of this paper, namely they correspond to interfaces that have *fixed inputs* (A) and *outputs* (B), e.g. as seen in (1).

The category Poly has all small limits. Suppose given a small category J and functor $p: J \rightarrow \text{Poly}$, and for each j , let p^j denote the corresponding polynomial. The limit $\lim_{j \in J} p$ has positions given by the limit $\lim_{j \in J} p^j(1)$ of positions, and for each such position $(i^j)_{j \in J}$, where $i^j \in p^j(1)$, the set of directions there is given by the colimit $\text{colim}_{j \in J} p_{ij}^j$ of directions.

We note two orthogonal factorization systems on Poly: (epi/mono) and (vertical/cartesian). The first is straightforward (e.g. epimorphisms of polynomials are surjective on positions and injective on directions). More interestingly, the functor $p \mapsto p(1)$ is a monoidal $*$ -bifibration in the sense of [Shu08, Definition 12.1]. Indeed, if $B = p(1)$ and we have a function $f: A \rightarrow B$, we can take the pullback of polynomials

$$\begin{array}{ccc} A \times_B p & \xrightarrow{\text{cart}_f} & p \\ \downarrow & \lrcorner & \downarrow \\ A & \xrightarrow{f} & B \end{array}$$

Thus we obtain a fibration $\text{Poly} \rightarrow \text{Set}$, with its attendant vertical/cartesian factorization system. Moreover each functor $f^*: \text{Poly}_B \rightarrow \text{Poly}_A$ has both a left adjoint $f_!$ and a right adjoint f_* , and both $f_!$ and f_* interact well with \otimes . In fact, identifying $\text{Poly}_A^{\text{op}}$ with Set^A , the functors $\text{Set}^I \rightarrow \text{Set}^J$ arising from multivariate polynomials $I \xleftarrow{f} E \xrightarrow{g} B \xrightarrow{h} J$ as in [GK12] can be represented using the $*$ -bifibration structure, namely as $(h_* g_! f^*)^{\text{op}}$.

2.3 Adjunctions with Set and Set^{op}

It is useful to note that Poly contains two copies of Set and a copy of Set^{op}, namely as the constant polynomials A , the linear polynomials Ay , and the representables y^A . Indeed there is an adjoint quadruple and an adjoint pair as follows, labeled by where they send objects $A \in \text{Set}, p \in \text{Poly}$:

$$\begin{array}{ccc} \text{Set} & \begin{array}{c} \xleftarrow{p(0)} \\ \xrightarrow{A} \\ \xleftarrow{p(1)} \\ \xrightarrow{Ay} \end{array} & \text{Poly} \end{array} \quad \text{Set}^{\text{op}} \begin{array}{c} \xrightarrow{y^A} \\ \xleftarrow{\Gamma p} \end{array} \text{Poly} . \quad (6)$$

All of the functors out of Set and Set^{op} shown in (6) are fully faithful, and the rightmost adjoint $p \mapsto p(0)$ preserves coproducts. The functor Γ is given by *global sections*: $\Gamma p := \text{Poly}(p, y) \cong \prod_{i \in p(1)} p_i$.

For each $A \in \text{Set}$ the functor $\text{Poly} \rightarrow \text{Set}$ given by $q \mapsto q(A)$ has a left adjoint, namely $B \mapsto By^A$; we saw this for the cases $A \cong 0, 1$ in (6). Using $p := y^A$ and the Yoneda lemma, this generalizes to a two-variable adjunction $\text{Set} \times \text{Poly} \rightarrow \text{Poly}$:

$$\text{Poly}(Ap, q) \cong \text{Poly}(p, q^A) \cong \text{Set}(A, \text{Poly}(p, q)). \quad (7)$$

2.4 Monoidal structures on Poly

We have already mentioned two monoidal structures on Poly, namely coproduct $(+, 0)$ and product $(\times, 1)$. They are given by the following formulas:

$$p + q = \sum_{i \in p(1)} y^{p_i} + \sum_{j \in q(1)} y^{q_j} \quad \text{and} \quad p \times q = \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i + q_j}. \quad (8)$$

These form a distributive category. The product monoidal structure is closed—Poly is cartesian closed—and we denote this closure operation by exponentiation:

$$q^p \cong \prod_{i \in p(1)} q \circ (p_i + y). \quad (9)$$

Thus for example $(y^2 + 3y + 2)^{y^5 + y^4} \cong ((5 + y)^2 + 3(5 + y) + 2) \cdot ((4 + y)^2 + 3(4 + y) + 2)$. The constant-polynomials functor $\text{Set} \rightarrow \text{Poly}$ is cartesian closed.

In terms of bundles, the coproduct is given by disjoint union, and product is given by adding fibers (though the formula is reminiscent of adding fractions):

$$\left(\begin{array}{c} E \\ \downarrow \\ B \end{array} \right) + \left(\begin{array}{c} E' \\ \downarrow \\ B' \end{array} \right) \cong \left(\begin{array}{c} E + E' \\ \downarrow \\ B + B' \end{array} \right) \quad \left(\begin{array}{c} E \\ \downarrow \\ B \end{array} \right) \times \left(\begin{array}{c} E' \\ \downarrow \\ B' \end{array} \right) \cong \left(\begin{array}{c} E \times B' + B \times E' \\ \downarrow \\ B \times B' \end{array} \right)$$

In terms of forests, coproduct (undrawn) is given by disjoint union and product is given by multiplying the roots and adding the leaves. Here is a picture of $(y + 1)(y + 2) \cong y^2 + 3y + 2$:

$$\begin{array}{|c|} \hline \uparrow \\ \hline \bullet \\ \hline \end{array} \times \begin{array}{|c|} \hline \uparrow \\ \hline \bullet \bullet \\ \hline \end{array} \cong \begin{array}{|c|} \hline \vee \uparrow \uparrow \uparrow \uparrow \\ \hline \bullet \bullet \bullet \bullet \\ \hline \end{array}$$

There are two more monoidal structures on Poly; one is symmetric and is denoted (\otimes, y) , and the other is not symmetric and is denoted (\circ, y) . We first discuss \otimes . In terms of polynomials, it is given by the *Dirichlet product*¹

$$p \otimes q = \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p_i q_j}, \quad (10)$$

which we invite the reader to compare with \times from (8). For example $(y^3 + y) \otimes (y^2 + y^0) \cong y^6 + y^2 + 2y^0$. Like \times , the Dirichlet product \otimes distributes over $+$. In terms of bundles, Dirichlet product is straightforward:

$$\left(\begin{array}{c} E \\ \downarrow \\ B \end{array} \right) \otimes \left(\begin{array}{c} E' \\ \downarrow \\ B' \end{array} \right) \cong \left(\begin{array}{c} E \times E' \\ \downarrow \\ B \times B' \end{array} \right).$$

¹The reason for the name *Dirichlet* is that if one replaces polynomials with Dirichlet series by reversing each summand y^A to A^y , the result is the usual product. For example

$$(3^y + 2^y) \times (4^y + 0^y) \cong 12^y + 8^y + 2 \cdot 0^y$$

In terms of forests, one multiplies roots and for each pair, multiplies the leaves:

$$\boxed{\begin{array}{c} \nearrow \nearrow \nearrow \\ \bullet \end{array}} \uparrow \boxed{\begin{array}{c} \nearrow \nearrow \\ \bullet \end{array}} \bullet \cong \boxed{\begin{array}{c} \nearrow \nearrow \nearrow \nearrow \nearrow \\ \bullet \end{array}} \bullet \bullet$$

The Dirichlet monoidal structure is closed as well and its formula is similar to that in (9). We denote this closure operation (internal hom) using brackets:

$$[p, q] \cong \prod_{i \in p(1)} q \circ (p_i y). \quad (11)$$

Thus for example $[y^5 + y^4, y^2 + 3y + 2] \cong ((5y)^2 + 3(5y) + 2) \cdot ((4y)^2 + 3(4y) + 2)$.

The last monoidal structure we discuss, (\circ, y) , was already used above in (??). It is the usual composition of polynomials, both algebraically and as functors; e.g. $(y^2 + y) \circ (y^3 + 1) \cong y^6 + 3y^3 + 2$. Thinking of p as a functor, its evaluation at a set A is $p \circ A$.

The most computationally useful formula for $p \circ q$ is probably the following:

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p_i} \sum_{j \in q(1)} \prod_{e \in q_j} y. \quad (12)$$

In terms of forests, $p \circ q$ is obtained by adding up all ways to adjoin trees in q to leaves in p . For example, here is $(y^2 + y) \circ (y^3 + 1)$:

$$\boxed{\begin{array}{c} \nearrow \nearrow \\ \bullet \end{array}} \bullet \circ \boxed{\begin{array}{c} \nearrow \nearrow \nearrow \\ \bullet \end{array}} \bullet \cong \text{sum of 5 forests} \quad (13)$$

More precisely, the monoidal operation \circ collapses the trees in (13) to mere corollas:

$$\boxed{\begin{array}{c} \nearrow \nearrow \\ \bullet \end{array}} \bullet \circ \boxed{\begin{array}{c} \nearrow \nearrow \nearrow \\ \bullet \end{array}} \bullet \cong \boxed{\begin{array}{c} \nearrow \nearrow \nearrow \nearrow \nearrow \\ \bullet \end{array}} \bullet \bullet$$

The composition product \circ is *duoidal* over \otimes in the sense that there is a natural map

$$(p_1 \circ p_2) \otimes (q_1 \circ q_2) \rightarrow (p_1 \otimes q_1) \circ (p_2 \otimes q_2), \quad (14)$$

satisfying the usual axioms. Both $+$ and \times commute with \circ on the left

$$(pq + r) \circ s \cong (p \circ s)(q \circ s) + (r \circ s).$$

2.5 Comonoids for \circ are categories

A comonoid in the (nonsymmetric) monoidal category (Poly, \circ, y) is a tuple (C, ϵ, δ) , where C is a polynomial,² and $\epsilon: C \rightarrow y$ and $\delta: C \rightarrow C \circ C$ are morphisms of polynomials, such that the usual diagrams commute. Using (14), we can lift the Dirichlet product on polynomials to a monoidal structure (\otimes, y) on comonoids.

One of the most surprising aspects of Poly is that the comonoids for \circ —polynomial comonads on Set —are categories in the usual sense! This requires a calculation (see

²We use upper case to denote the polynomials that underlie comonoids.

Theorem 2.6), though it can be visualized using tree composition (13). Sums and Dirichlet products of comonoids correspond to coproducts and products of categories, respectively.

Note that morphisms of comonoids correspond not to functors but to *cofunctors*, first defined in [HM93]; see also [Agu97]. This notion is not well-enough known, so we recall it.

Definition 2.5 (Cofunctor). Let C and D be categories. A *cofunctor* $F: C \rightarrow D$ consists of

1. a function $F_0: C_0 \rightarrow D_0$ on objects and
2. a function $F^\sharp: C_0 \times_{D_0} D_1 \rightarrow C_1$ backwards on morphisms,

satisfying the following conditions:³

- i. $F^\sharp(c, \text{id}_{F_0 c}) = \text{id}_c$ for any $c \in \text{Ob}(C)$;
- ii. $F_0 \text{cod } F^\sharp(c, g) = \text{cod } g$ for any $c \in \text{Ob}(C)$ and $g: F_0(c) \rightarrow \text{cod}(g)$ in D ;
- iii. $F^\sharp(\text{cod } F^\sharp(c, g_1), g_2) \circ F^\sharp(c, g_1) = F^\sharp(c, g_2 \circ g_1)$ for composable arrows g_1, g_2 out of $F_0 c$.

Theorem 2.6 (Ahman-Uustalu [AU16]). The following categories are equivalent:

1. the category Cat^\sharp of categories and cofunctors;
2. the category $\text{Comon}(\text{Poly})$ of comonoids in (Poly, \circ, y) and comonoid morphisms.

Example 2.7 (Contractible groupoids, Sy^S). Let S be a set; the contractible groupoid on S is the category with objects S and a unique morphism $s \rightarrow s'$ for each $s, s' \in S$. It corresponds to the comonoid with carrier Sy^S and counit $Sy^S \rightarrow y$ given by evaluation. In other words it is the comonad $\text{Set} \rightarrow \text{Set}$ arising from the exponential adjunction for the set S . It is often called the *store comonad* in functional programming.

Remark 2.8. In <https://www.youtube.com/watch?v=tW6HYnqn6eI>, Richard Garner explains that for any comonoids C, D , the (C, D) -bimodules in Poly are precisely the *parametric right adjoints* $D\text{-Set} \rightarrow C\text{-Set}$ between the copresheaf categories.

In [AU13] it was shown that the comonoid arising from a distributive law $D \circ C \rightarrow C \circ D$ between comonoids in Poly recovers the Zappa-Szép product [Zap40] of monoids when the C, D are themselves monoids.

The point is that comonoids in Poly unexpectedly recover many important notions.

3 Polynomials and mode-dependent dynamics

We next discuss how structures available in Poly describe phenomena in dynamical systems.

³ The cofunctor laws written in diagram form are as follows:

$$\begin{array}{ccccc}
 C_0 \times_{D_0} D_0 & \xrightarrow{\cong} & C_0 & C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} C_1 \xrightarrow{\text{cod}} C_0 & C_0 \times_{D_0} D_1 \times_{D_0} D_1 & \xrightarrow{\circ_{D_0}} C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} C_1 \\
 \text{id}_D \downarrow & \text{(i)} & \downarrow \text{id}_C & \pi_2 \downarrow & \downarrow F_0 & F^\sharp \downarrow & \text{(iii)} & \uparrow \circ_C \\
 C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} C_1 & D_1 & \xrightarrow{\text{cod}} D_0 & C_1 \times_{D_0} D_1 & \xrightarrow{\cong} C_1 \times_{C_0} C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} C_1 \times_{C_0} C_1
 \end{array}$$

3.1 Dynamical systems in Poly

By a fixed-interface (A, B) -dynamical system, we mean a *Moore machine*, i.e. a function $r: S \rightarrow B$ (called *readout*), and a function $u: A \times S \rightarrow S$ (called *update*). Given an initial state $s_0 \in S$, a Moore machine lets us transform any stream (a_0, a_1, \dots) of A 's into a stream of B 's by repeatedly updating the state:

$$s_{n+1} = u(a_n, s_n), \quad b_n = r(s_n).$$

Proposition 3.1. *Let S, A, B be sets. The following are equivalent:*

1. *Moore machines with inputs A , outputs B ,*
2. *coalgebras for the polynomial functor By^A ,*
3. *morphisms in Poly of the form $Sy^S \rightarrow By^A$.*

The second and third perspectives easily generalize to replacing By^A with an arbitrary polynomial. We prefer the third because it allows us to remain within the category Poly , which has such abundant structure. Recall from Example 2.7 that Sy^S can be given the structure of a comonoid in (Poly, \circ, y) , corresponding under Theorem 2.6 to the contractible groupoid on S .

Definition 3.2. A *mode-dependent dynamical system* consists of a comonoid (C, ϵ, δ) in (Poly, \circ) together with a morphism $f: C \rightarrow p$ for some polynomial p . Here C is called the *state system* p is called the *interface* and f is called the *dynamics*.

Note that given such a morphism $f: C \rightarrow p$, the comonoid structure on C gives us a canonical morphism $\delta^{n-1}: C \rightarrow C^{\circ n}$ for each n , where $\delta^{-1} = \epsilon$ and $\delta^0 = \text{id}$. Since \circ is monoidal, we also have a map $f^{\circ n}: C^{\circ n} \rightarrow p^{\circ n}$, and composing we obtain

$$C \rightarrow C^{\circ n} \rightarrow p^{\circ n}.^4$$

Thus each $i \in C(1)$ is endowed with an element of $p^{\circ n}(1)$, which by (12) can be understood as a length- n *strategy*

$$p^{\circ n}(1) \cong \sum_{i_1 \in p(1)} \prod_{d_1 \in p_{i_1}} \sum_{i_2 \in p(1)} \prod_{d_2 \in p_2} \cdots \sum_{i_n \in p(1)} \prod_{d_n \in p_n} 1.$$

It is a choice of a position (move by ‘player’) in $i_1 \in p(1)$, and for every direction there (move by ‘opponent’) $d_1 \in p_{i_1}$, a choice of position $i_2 \in p(1)$, etc. Thus a sort of *game* is inherent in the dynamical system itself; it would be interesting to explore a relationship between this and open economic game theory [Gha+16].

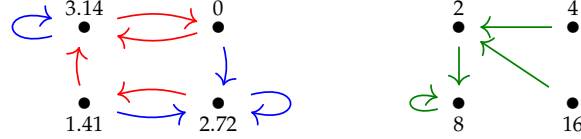
But the map $C \rightarrow p^{\circ n}$ does not only give a mapping on positions; it says that for every n choices of directions—each dependent on the last—in p , there is a choice of direction, i.e. morphism, in the comonoid/category C . Thus the history of play is encoded as a morphism in C . In the case of coalgebras, where $C = Sy^S$ is simply a contractible groupoid, there is no information encoded in this history of play, except for its final destination.

⁴For those readers who are more accustomed to coalgebras, note that one can take the limit of the on-positions functions $C(1) \rightarrow p^{\circ n}(1)$, as n increases; this induces the usual map from $C(1)$ to the terminal coalgebra of p . In Poly one represents this by a right adjoint $\text{Poly} \rightarrow \text{Comon}(\text{Poly})$ to the forgetful functor. That is, $C \rightarrow p$ induces a comonoid morphism $C \rightarrow \text{Cof}(p)$ to the cofree comonoid on p , which itself is given by the limit $1 \leftarrow y \cdot p(1) \leftarrow y \cdot p(y \cdot p(1)) \leftarrow \cdots$ in Poly ; its set of positions $\text{Cof}(p)(1)$ is again the terminal coalgebra on p .

3.2 Products of interfaces

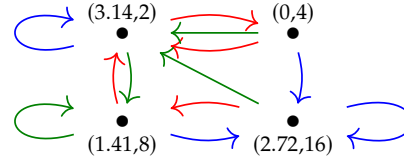
The product of polynomials allows one to overlay two different interfaces on the same state system. That is, given dynamical systems $C \rightarrow p$ and $C \rightarrow q$, there is a unique dynamical system $C \rightarrow pq$. This is quite useful for dynamical systems, as we now show.

Example 3.3. Consider two four-state dynamical systems $4y^4 \rightarrow \mathbb{R}_y^{\{r,b\}}$ and $4y^4 \rightarrow \mathbb{R}_y^{\{g\}}$, each of which gives outputs in \mathbb{R} ; we think of r, b, g as red, blue, and green, respectively. We can draw such morphisms as labeled transition systems, e.g.



Each bullet refers to a state, is labeled by its output position in \mathbb{R} , and has a unique emanating arrow for each sort of input (red and blue, or green), indicating how that state is updated upon encountering said input.

The universal property of products provides a unique way to put these systems together to obtain a morphism $4y^4 \rightarrow (\mathbb{R}_y^{\{r,b\}} \times \mathbb{R}_y^{\{g\}}) = (\mathbb{R}^2)_y^{\{r,b,g\}}$. With the examples above, it looks like this:



Thus the intuitively obvious act of overlaying these dynamical systems falls out of the mathematics, in particular the universal property of products \times in Poly. This works for non-monomial (context-dependent) interfaces as well.

3.3 Wiring diagrams and mode-dependence

The Dirichlet product (10) of polynomials and comonoids allows us to juxtapose dynamical systems in an environment. That is, given dynamical systems $C_1 \rightarrow p_1$ and $C_2 \rightarrow p_2$, we can form a new dynamical system $(C_1 \otimes C_2) \rightarrow (p_1 \otimes p_2)$.

Example 3.4 (Wiring diagrams). Suppose given a wiring diagram such as that in (1); as mentioned in (2), the interfaces of the controller and plant are the polynomials By^C and Cy^{AB} , and that of the total system is Cy . All of these are monomials, meaning that the directions do not depend on the positions; this allows us to think of positions as outputs and directions as inputs, drawn on the right and left of boxes respectively. The wiring diagram (1) itself is syntax for a morphism

$$By^C \otimes Cy^{AB} \rightarrow Cy^A. \quad (15)$$

On positions the required map $BC \rightarrow C$ is the projection, and on directions the required map $BCA \rightarrow CAB$ is the obvious symmetry.

Example 3.5 (Mode-dependent wiring diagrams). In (3) we depicted a company C changing its supplier of widgets W , based on C 's internal state. The company was shown with

no output wires, but in fact it has two positions corresponding to choosing supplier 1 or supplier 2. Let's redraw it to emphasize its change of position:



The company has interface $2y^W$, and the each supplier has interface Wy ; let's take the total system interface (undrawn) to be the closed system y . Then this mode-dependent wiring diagram is just a map $2y^W \otimes Wy \otimes Wy \rightarrow y$. Its on-positions function $2W^2 \rightarrow 1$ is uniquely determined, and its on-directions function $2W^2 \rightarrow W$ is the evaluation. In other words, the company's position determines which supplier from which it receives widgets.

Similarly we could say that the person in (4) has interface $2y$, the units have interfaces Xy and y^X respectively, and the whole system is closed; that is, the diagram represents a morphism $2y \otimes Xy \otimes y^X \rightarrow y$. We did not mention but need that unit B has a default value, say $x_0 \in X$, for when its input wire is unattached. The morphism $2Xy^X \rightarrow y$ is uniquely determined on positions, and on directions it is given by cases $(1, x) \mapsto x_0$ and $(2, x) \mapsto x$.

References

- [Agu97] Marcelo Aguiar. *Internal categories and quantum groups*. Cornell University, 1997.
- [AU13] Danel Ahman and Tarmo Uustalu. "Distributive laws of directed containers". In: *Progress in Informatics* 10 (2013), pp. 3–18.
- [AU16] Danel Ahman and Tarmo Uustalu. "Directed Containers as Categories". In: *EPTCS 207, 2016*, pp. 89–98 (2016). eprint: [arXiv:1604.01187](https://arxiv.org/abs/1604.01187).
- [Gha+16] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. "Compositional game theory". In: *Proceedings of Logic in Computer Science (LiCS) 2018* (2016).
- [GK12] Nicola Gambino and Joachim Kock. "Polynomial functors and polynomial monads". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154.1 (Sept. 2012), pp. 153–192.
- [Hed18] Jules Hedges. *Limits of bimorphic lenses*. 2018. eprint: [arXiv:1808.05545](https://arxiv.org/abs/1808.05545).
- [HM93] Philip J Higgins and Kirill CH Mackenzie. "Duality for base-changing morphisms of vector bundles, modules, Lie algebroids and Poisson structures". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 114. 3. Cambridge University Press. 1993, pp. 471–488.
- [Shu08] Michael Shulman. "Framed bicategories and monoidal fibrations". In: *Theory and Applications of Categories* 20 (2008), Paper No. 18, 650–738.
- [SM20] David I. Spivak and David Jaz Myers. *Dirichlet Polynomials form a Topos*. 2020. eprint: [arXiv:2003.04827](https://arxiv.org/abs/2003.04827).

- [ST17] David I Spivak and Joshua Tan. “Nesting of dynamical systems and mode-dependent networks”. In: *Journal of Complex Networks* 5.3 (2017), pp. 389–408.
- [Zap40] Guido Zappa. “Sulla costruzione dei gruppi prodotto di due dati sottogruppi permutabili tra loro”. In: *Atti Secondo Congresso Un. Mat. Ital., Bologna. 1940*, pp. 119–125.