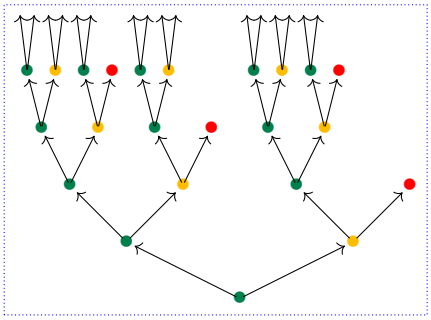
Operational Semantics for Distributed Systems



David I. Spivak Nelson Niu

(Last updated: December 4, 2020)

This page intentionally left blank.

To André Joyal

Preface

Acknowledgments

Thanks go to Neil Ghani, Joachim Kock, Sophie Libkind, David Jaz Myers, Richard Garner, Todd Trimble.

Contents

Chapter 1

Polynomial functors and dynamics

It is a treasury box!
Full of unexpected connections!
It is fascinating!
I will think about it.

–André Joyal, Summer 2020,
personal communication.

1.1 Introduction

In this chapter we will investigate a remarkable category called **Poly**. We will see its intimate relationship with both dynamic processes, data storage and transformations, and decision-making. But our story begins with something quite humble: middle school algebra.

$$y^2 + y + 1 \qquad \textit{polynomial} \tag{1.1}$$

All our polynomials will involve one variable, y, chosen for reasons we'll explain soon. Polynomials in one variable can be depicted as a set of mini-trees:

arena (1.2)

More technically, mini-trees are called *corollas*, so what we're calling an *arena*, in this chapter, is a set of corollas.

Intuitively, one might think of each corolla as representing a *decision*. Associated to every decision is a set of *options*. The three decisions we exhibit in Eq. (1.2) are particularly interesting; they respectively have two options, one option, and no options. Having two options is familiar from life—it's the classic yes/no decision—as well as from Claude Shannon's Information Theory. Having one option is also familiar theoretically and in life: "sorry, ya just gotta go through it." Having no options is when you actually don't get through it: it's "the end". While the corollas $1, y$, and y^2 are

each interesting as decisions, their sum $y^2 + y + 1$ has very little theoretical interest; it's just a first example.

In a polynomial like $42y^3 + 17y$, the pure power term y^3 has a coefficient of 42 next to it, so the corolla with three leaves shows up 42 times[1] in the associated arena. Intuitively, there are 42 situations in which you have a decision with three options. When you put all those decisions together, you have the *arena* associated to p.

Here is a table of terminology. The first column is book-wide terminology, the second is algebraic terminology about polynomials as in (1.1), the third is about the pictures of corollas as in (1.2), and the fourth column is intuitive terminology.

Terminology

Arena	**Polynomial** p	**Decision Trees**	**Intuition**
Set of positions	$p(1)$	Set of roots	Set of situations
Interface in position i	Pure power summand $y^{p[i]}$	Corolla	Decision
Direction $c \in p[i]$	Element of $p[i]$	Leaf	Possibility

(1.3)

Exercise 1.4. Consider the polynomial $p \coloneqq 2y^3 + 2y + 1$ and the associated arena.

1. Draw the arena whose name is p as a set of corollas.
2. How many roots does this arena have?
3. How many decisions does this represent?
4. For each corolla in the arena, say how many leaves it has.
5. For each decision, how many options does it have?

Now referring instead to the polynomial $q \coloneqq y^{\mathbb{N}} + 4y$:

6. Does the polynomial q have a pure-power summand y^2?
7. Does the polynomial q have a pure-power summand y?
8. Does the polynomial q have a pure-power summand $4y$? ◊

Remark 1.5. Though a polynomial is a *functor* and an arena is a *dependent set*, they are so closely related that we do not make a distinction between a polynomial p and its arena; they are two different syntaxes for the same object.

Exercise 1.6. If you were a suitor choosing the arena you love, aesthetically speaking, which would strike your interest? Answer by circling the associated polynomial:

1. $y^2 + y + 1$
2. $y^2 + 3y^2 + 3y + 1$
3. y^2
4. $y + 1$
5. $(\mathbb{N}y)^{\mathbb{N}}$

[1] In standard font, 42 represents the usual natural number. In sans serif font 42 represents the set $\mathsf{42} = \{1, 2, \ldots, 42\}$ with 42 elements.

6. Sy^S
7. $y^{100} + y^2 + 3y$
8. Your poly's name p here.

Any reason for your choice? Draw a sketch of your arena. ◊

Before we can really get into this story, let's summarize where we're going: polynomials are going to have really surprising applications to dynamics, data, and decision. Remember, we spoke superlatively of **Poly** at the beginning of this chapter,

The category of polynomial functors is a jackpot. Its beauty flows without bound

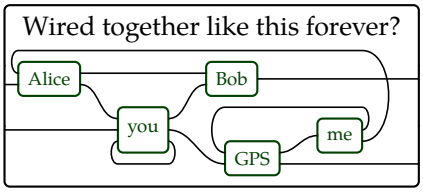
and we have not yet begun to deliver. So let's introduce some of the applications and mathematics to come.

1.1.1 Dynamical systems

Throughout this book, we've seen dynamical systems—machines of various sorts—which have an internal state that can be read out to other systems, as well as updated based on input received from other systems. In the context of this chapter, we'll be looking mainly at deterministic systems, but with a lot of interesting new options:

1. The interface of the system can change shape through time.
2. The wiring diagram connecting a bunch of systems can change in time.
3. One can speed up the dynamics of a system.
4. One can introduce "effects", i.e. as defined by monads on **Set**.
5. The dynamical systems on any interface form a topos.

To give some intuition for the first two, imagine yourself as a system, wired up to other systems. You have some input ports: your eyes, your ears, etc., and you have some output ports: your speech, your gestures, etc. And you connect with other systems: your family, your colleagues, the GPS of your phone, etc.



(1.7)

We wrote a little question for you at the top of the diagram. Isn't there something a little funny about wiring diagrams? Maybe for old-fashioned machines, you would wire things together once and they'd stay like that for the life of the machine. But my phone connects to different wifi stations at different times, I drop my connection to Alice for weeks at a time, etc. So wiring diagrams should be able to change in time; **Poly** will let us do that.

Example 1.8. Here are some familiar circumstances where we see wiring diagrams changing in time.

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

   In materials science the Young's modulus accounts for how much force can be transferred across a material as its endpoints are pulled apart. When the material breaks, the two sides can no longer feel evidence of each other. Thinking of pulling as sending a signal (a signal of force), we might say that the ability of internal entities to send signals to each other—the connectivity of the wiring diagram—is being measured by Young's modulus. It will also be visible within **Poly**.

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

   The company can get widgets either from supplier 1 or supplier 2; we could imagine this choice is completely up to the company. The company can decide based on the quality of widgets it has received in the past, i.e. when the company gets a bad widget, it updates an internal variable, and sometimes that variable passes a threshold making the company switch states. Whatever its strategy for deciding, we should be able to encode it in **Poly**.

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

   Have you ever assembled something? Your internal states dictate the connection pattern of some other things. We can say this in **Poly**.

All of the examples discussed here will be presented in some detail once we have the requisite mathematical theory Examples 1.154 to 1.156.

Exercise 1.9. Think of another example where systems are sending each other information, but where the sort of information or who it's being sent to or received from can change based on the states of the systems involved. You might have more than two, say $\mathbb{R}$-many, different wiring patterns in your situation. ◊

But there's more that's intuitively wrong or limiting about the picture in (1.7). Ever notice how you can change how you interface with the world? Sometimes I close my eyes, which makes that particular way of sending me information inaccessible: that port vanishes and you need to use your words. Sometimes I'm in a tunnel and my phone can't receive GPS. Sometimes I extend my hand to give or receive an object from another person, but sometimes I don't. My ports themselves change in time. Sometimes I even use my output ports to determine the wiring pattern of other machines. We will be able to say all this using **Poly**.

And there's even more that's wrong with the above description. Namely, when I move my eyes, that's actually something you can see—e.g. whether I'm looking at you. When I turn around, I see different things, and *you can notice I'm turned around*! When I use my muscles or mouth to express things, my very position changes: my tongue moves, my body moves. So my output is actually achieved by changing position. The model in **Poly** will be able to express this too.

Example 1.10. Imagine a million little eyeballs, each of which has a tiny brain inside it, all together in a pond of eyeballs. All that an individual eyeball e can do is open and close. When e is open, it can make some distinction about all the rest of the eyeballs in view: maybe it can count how many are open, or maybe it can see whether just one certain eyeball e' is open or closed. But when e is closed, it can't see anything; whatever else is happening, it's all the same to e. All it can do in that state is process previous information.

Each eyeball in this system will correspond to the polynomial y^n+y, which intuitively consists of two situations, one with n-many options, and the other with only one option.

The point is, however, is that the other eyeballs can tell if e is opened or closed. We can imagine some interesting dynamics in this system, e.g. waves of openings or closings sweeping over the group, a ripple of openings expanding through the pond.

Talk about real-world applications!

Hopefully you now have an idea of what we mean by mode-dependence: interfaces and wiring diagrams changing in time, based on the states of all the systems involved. We'll see that **Poly** speaks about mode-dependent systems and wiring diagrams in this sense.

But **Poly** is very versatile in its applications. Next we'll show how it relates to information storage and translation.

1.1.2 Data

Data is information, maybe thought of as quantized into atomic pieces, but where these atomic pieces are somehow linked together according to a conceptual structure. When a person or organization uses certain data repeatedly, they often find it useful to put their data in a database. This requires organizing the little pieces into a conceptual structure. So when you hear "database", just think of it as a conceptual structure filled with examples.

To fix a mental image, let's say that you need to constantly look up employees, what department they're in, who the admin person is for that department, who their manager is, etc. Here's an associated database

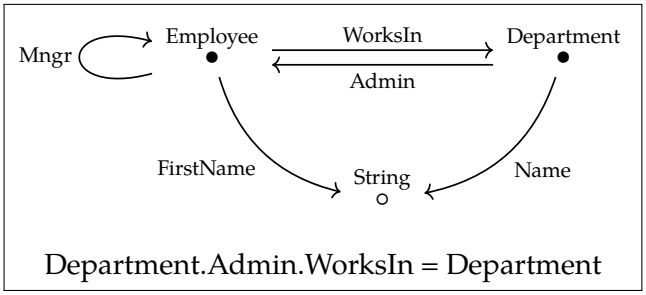
Employee	FirstName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Carla	102	3

Department	Name	Admin
101	Sales	1
102	IT	3

(1.11)

We can see it as being associated to the following conceptual scheme, also called a *schema*:

$\mathcal{C} :=$



(1.12)

The equation at the bottom says that for any department d, if you ask for the admin person and see which department they work in, it's required to be d.

What's called $\mathcal{C}$ in (1.12) is a *finitely presented category*, and the database instance presented in (1.11) correspond to a functor $I : \mathcal{C} \to \mathbf{Set}$; it sends the object Employee $\in \mathcal{C}$ to the set $I(\text{Employee}) := \{1, 2, 3\}$. It sends the morphism Mngr: Employee $\to$ Employee to the function $I(\text{Mngr}) : \{1, 2, 3\} \to \{1, 2, 3\}$ given by $I(1) = 2$, $I(2) = 2$, and $I(3) = 3$. A functor $\mathcal{C} \to \mathbf{Set}$ is called a *copresheaf on* $\mathcal{C}$. So the story of database schemas and their data can be based on the story of categories and their copresheaves.

There's a very important thing that we do with databases: we query them. We ask them questions like "tell me everyone that's either the admin person of the Math department or their manager".

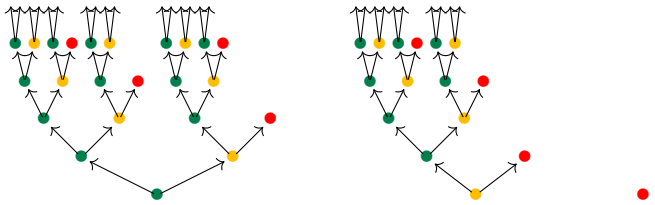
```
FOR    d: Department, e: Employee
WHERE  Name(d)="Math" AND
       (e=Admin(d) OR e=Mngr(Admin(d)))
RETURN FirstName(e)
```

This sort of question is formally called a "union of conjunctive queries". We will see this sort of query is intimately connected with **Poly**.

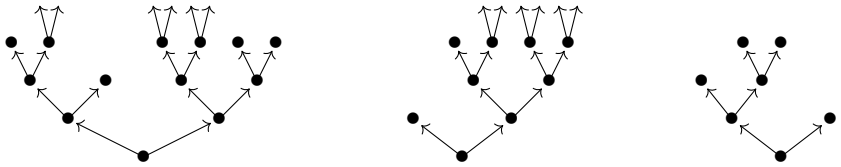
We will also see how databases can be conceived in terms of dynamical systems.

1.1.3 Decisions

Consider the following three trees, the first two are infinite (though that's hard to draw):



These are patterned examples—and we'll understand what this pattern is more clearly in **??**—of what we will call *decision streams*. Decision streams form the objects in a category that also includes the following level-3 abbreviations of binary decision streams (the third of which is a finite stream):

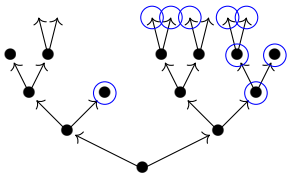


The set of such decision streams in fact form the objects of a category with very nice properties (it's a topos), which we call $\mathbf{Sys}(y^2+1)$. The idea is that every corolla in these diagrams has either two options, corresponding to y^2, or no options, corresponding to $1 = y^0$.

Exercise 1.13.

1. Draw a level-3 abbreviation of a decision stream of type $y^2 + y^0$.
2. Draw a level-4 abbreviation of a decision stream of type y.
3. Draw a level-3 abbreviation of a decision stream of type $\mathbb{N}y^2$ by labeling every node with a natural number. ◊

But decisions aren't just about choosing; they're also about trying to accomplish something. The logic of accomplishment is exceptionally rich in this setting. We will concentrate on what we call a *win condition*, which is a subgraph of the decision stream with the property that if n is winning node, then any child of n is also a winning node.



More formally, these are called *sieves*; they form the elements of a logical system called a Heyting algebra: you can take any two sieves and form the intersection or union (which correspond to AND and OR), or even things like implication, negation, and existential and universal quantification. This will give us a calculus of win-conditions for any sort p of decision stream.

1.1.4 Implementation

Everything we talk about can actually be implemented in a computer without much difficulty, at least if you have access to a language that supports dependent types. We will continue to use Agda.

What we have been calling polynomials—things like $y^2 + 2y + 1$—are often called *containers* in the computer science literature. A container consists of a type S, usually called the type of *shapes*, and a type $P(s)$ for each term $s : S$, called the type of positions in shape s. It's mildly unfortunate that the names clash with our own: for us a container-shape is a position and a container-position is a distinction.

Luckily, the agda code is pretty easy to understand.

```
record Arena : Set where  -- a arena consists of
  field                   -- two fields
    decn : Set            -- one called decn, a set, and
    posy : pos -> Set     -- one called posy,
                          -- a set that depends on decn
```

1.1.5 Mathematical theory

The applications of **Poly** are quite diverse and interesting, including dynamics, data, and decisions. However it is how the mathematics of **Poly** supports these applications that is so fantastic. For experts, here are some reasons for the excitement.

Proposition 1.14. **Poly** is a biCartesian closed category, meaning it has sums, products, and exponential objects. It supports the simply typed lambda calculus.

In fact **Poly** has infinite coproducts, infinite products, and is completely distributive. We'll explain all of this in **??**; for now we continue with the highlights.

Proposition 1.15. Beyond the coCartesian and Cartesian monoidal structures $(0, +)$ and $(1, \times)$, the category **Poly** has two additional monoidal structures, denoted $(y, \otimes)$ and $(y, \circ)$, which are together duoidal. Moreover $\otimes$ is also a closed monoidal structure and it distributes over $+$.

Proposition 1.16. **Poly** has an adjoint quadruple with **Set** and an adjoint pair with $\mathbf{Set}^{\mathrm{op}}$:

Each functor is labeled by where it sends $p \in \mathbf{Poly}$ or $A \in \mathbf{Set}$.

There's a lot we're leaving out of this summary, just so we can hit the highlights.

Proposition 1.17. The functor $\mathbf{Poly} \to \mathbf{Set}$ given by $p \mapsto p(1)$ is a monoidal fibration.

In fact it's a monoidal $*$-bifibration in the sense of [**shulman2008framed**]. But here's where things get really interesting.

Proposition 1.18 (Ahman-Uustalu)**.** Comonoids in $(\mathbf{Poly}, y, \circ)$ are categories (up to isomorphism).

Proposition 1.19. The category $\mathbf{Comon}(\mathbf{Poly})$ has finite coproducts and products, and coproducts in $\mathbf{Comon}(\mathbf{Poly})$ agree with those in **Cat**.

Proposition 1.20. Suppose that the category $\mathcal{C}$ corresponds under Proposition 1.18, to the comonoid $\mathscr{C}$. Then there is an equivalence of categories

$$\mathbf{Bimod}(\mathscr{C}, 0) \cong \mathcal{C}\text{-}\mathbf{Set}$$

between $(\mathscr{C}, 0)$-bimodules and $\mathcal{C}$-copresheaves.

We will use the convention that the comonoid $\mathscr{C}$ corresponds to category $\mathcal{C}$, and similarly for $\mathscr{D}$ and $\mathcal{D}$, etc.

Proposition 1.21 (Garner)**.** For any categories $\mathcal{C}$ and $\mathcal{D}$, there is an equivalence of categories

$$\mathbf{Bimod}(\mathscr{C}, \mathscr{D}) \cong \mathbf{pra}(\mathcal{C}\text{-}\mathbf{Set}, \mathcal{D}\text{-}\mathbf{Set})$$

between that of bimodules between comonoids in **Poly** and parametric right adjoints between copresheaf categories.

Proposition 1.22. For any category $\mathcal{C}$ the category of left $\mathscr{C}$-modules is equivalent to the category of functors $\mathcal{C} \to \mathbf{Poly}$,

$$\mathbf{Bimod}(\mathscr{C}, y) \cong \mathbf{Fun}(\mathcal{C}, \mathbf{Poly}).$$

Proposition 1.23. The functor $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$ has a right adjoint

$$\mathbf{Poly} \underset{u_-}{\overset{\mathscr{K}_-}{\underset{\longleftarrow}{\overset{\longrightarrow}{\Leftarrow}}}} \mathbf{Comon}(\mathbf{Poly}) \ ,$$

called the *cofree comonoid* construction. It is lax monoidal with respect to $\otimes$.

Proposition 1.24. The category $\mathbf{Comon}(\mathbf{Poly})$ has a third symmetric monoidal structure $(y, \otimes)$, and the functor $u_- \colon (\mathbf{Comon}(\mathbf{Poly}), y, \otimes) \to (\mathbf{Poly}, y, \otimes)$ is strong monoidal.

Proposition 1.25. For any polynomial p, the category

$$\mathbf{Sys}(p) \cong \mathscr{K}_p\text{-}\mathbf{Set}$$

of dynamical systems on p forms a topos.

Proposition 1.26. A morphism $p \to q$ of polynomials induces a pre-geometric morphism between their respective toposes

$$\mathbf{Sys}(p) \underset{\longleftarrow}{\overset{\longrightarrow}{\Rightarrow}} \mathbf{Sys}(q) \ .$$

If you skipped over any of that—or all of that—it'll be no problem whatsoever! We will get to everything in the current section, 1.1.5, over the next few chapters. To briefly summarize the results we have just seen, they say in particular that there is a full-fledged logic of dynamical systems inhabiting any interface (Proposition 1.25), that these logics can be combined and compared (Proposition 1.26), and that the whole story of dynamics carries a strong connection to database theory. There are many avenues for study, but we need to push forward.

Let's begin.

1.2 Introduction to Poly as a category

In this section, we will set down the basic category-theoretic story of **Poly**, so that we can have a firm foundation from which to speak about dynamical systems, decisions, and data. We begin with the category **Set** of sets and functions, and what is arguably the fundamental theorem of category theory, the Yoneda lemma.

1.2.1 Representable functors and the Yoneda lemma

Definition 1.27. Given any set S, we denote by $y^S : \mathbf{Set} \to \mathbf{Set}$ the functor that sends any set X to the set $X^S = \mathbf{Set}(S, X)$, and sends any function $h : X \to Y$ to the function $h^S : X^S \to Y^S$, the one that sends $g : S \to X$ to $g \fatsemi h : S \to Y$.

We refer to y^S as the functor *represented by* S.

The symbol y stands for Yoneda, for reasons we will get to in Lemma 1.36. For now, here are some pictures for your eyes to gaze at; they are the poly's corresponding to various representables:

(1.28)

Example 1.29. The functor that sends every set X to $X \times X$, and sends $h : X \to Y$ to $(h \times h) : (X \times X) \to (Y \times Y)$ is representable.

Exercise 1.30. For each of the following functors $\mathbf{Set} \to \mathbf{Set}$, say if it is representable or not; if it is, say what the representing set is.

1. The identity functor $X \mapsto X$.
2. The constant functor $X \mapsto 2$.
3. The constant functor $X \mapsto 1$.
4. The constant functor $X \mapsto 0$.
5. The functor $X \mapsto X^{\mathbb{N}}$.
6. The functor $X \mapsto 2^X$. ◊

Proposition 1.31. For any function $f : R \to S$, there is an induced natural transformation $y^f : y^S \to y^R$; on any set X the X-component $X^f : X^S \to X^R$ is given by sending $g : S \to X$ to $f \fatsemi g : R \to X$.

Exercise 1.32. Prove that for any function $f : R \to S$, what we said was a natural transformation in Proposition 1.31 really is natural. That is, for any function $h : X \to Y$, show that the following diagram commutes:

$$\begin{array}{ccc} X^S & \xrightarrow{h^S} & Y^S \\ {\scriptstyle X^f}\downarrow & ? & \downarrow{\scriptstyle Y^f} \\ X^R & \xrightarrow[h^R]{} & Y^R \end{array}$$

◊

Exercise 1.33. Let $X = \{a, b, c\}$. For each of the following sets R, S and functions $f : R \to S$, describe the X-component of, i.e. the function $X^S \to X^R$ coming from, the natural transformation $y^f : y^S \to y^R$.

1. $R = 5$, $S = 5$, $f = \mathrm{id}$. (Here you're supposed to give a function called $X^{\mathrm{id}_5} : X^5 \to X^5$.)
2. $R = 2$, $S = 1$, f is the unique function.
3. $R = 1$, $S = 2$, $f(1) = 1$.
4. $R = 1$, $S = 2$, $f(1) = 2$.
5. $R = 0$, $S = 5$, f is the unique function.
6. $R = \mathbb{N}$, $S = \mathbb{N}$, $f(n) = n + 1$. ◊

Exercise 1.34. Show that the construction in Proposition 1.31 is functorial

$$y^- : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}^{\mathbf{Set}}, \tag{1.35}$$

as follows.

1. Show that for any set S, we have $y^{\mathrm{id}_S} : y^S \to y^S$ is the identity.
2. Show that for any functions $f : R \to S$ and $g : S \to T$, we have $y^g \fatsemi y^f = y^{f \fatsemi g}$. ◊

Lemma 1.36 (Yoneda lemma)**.** Given a functor $F : \mathbf{Set} \to \mathbf{Set}$ and a set S, there is an isomorphism

$$F(S) \cong \mathsf{Nat}(y^S, F) \tag{1.37}$$

where Nat denotes the set of natural transformations. Moreover, (1.37) is natural in both S and F.

Sketch of proof. For any natural transformation $m : y^S \to F$, consider the component $m_S : S^S \to F(S)$. Applying it to the identity on S as an element of S^S, we get an element $m_S(\mathrm{id}_S) \in F(S)$.

For any element $a \in F(S)$, there is a natural transformation $m_a : y^S \to F$ whose component on X is the function $X^S \to F(X)$ given by sending $g : S \to X$ to $F(g)(a)$. In Exercise 1.38 we ask you to show that this is natural in X and that these two constructions are mutually inverse. □

Exercise 1.38. Whoever solves this exercise can say they've proved the Yoneda Lemma.

1. Show that for any $a \in F(S)$, the maps $X^S \to F(X)$ given as in Lemma 1.36 are natural in X.
2. Show that the two mappings from Lemma 1.36 are mutually inverse.
3. As a corollary of Lemma 1.36, show that $y^- : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}^{\mathbf{Set}}$ is fully faithful, in particular that there is an isomorphism $\mathsf{Nat}(y^S, y^T) \cong S^T$. ◊

1.2.2 Polynomials: sums of representables

We've seen that for any set A, the symbol y^A represents a functor **Set** $\to$ **Set**. We will generalize this by adding representable functors together to form polynomials. In some sense the name "polynomial" doesn't quite fit, because in algebra polynomials are often taken to be finite sums, whereas we will use sums that may be infinite. However, we are not the first by far to use this term.

So here's the deal. All of our polynomials will be polynomials in one variable, y; every other letter or number that shows up will represent a set.[2] For example, in the following crazy polynomial

$$p \coloneqq \mathbb{R}y^{\mathbb{Z}} + 3y^3 + Ay + \sum_{i \in I} Q_i y^{R_i + Q_i^2}, \tag{1.39}$$

$\mathbb{R}$ denotes the set of real numbers, $\mathbb{Z}$ denotes the set of integers, 3 denotes the set $\{1, 2, 3\}$, and A, I, Q_i, and R_i denote some arbitrary sets that should have already been defined in order for (1.39) to make sense.

The polynomials we already understand at this point are the pure-power polynomials y^A for some set A; they are representable functors $y^A \colon$ **Set** $\to$ **Set**. So to understand general polynomials like p, we just need to understand sums of functors. It will be useful to discuss products of functors at the same time. However, we haven't discussed sums and products of sets yet, so we need to do that first.

Dependent sums and products of sets. Let I be a set, and let X_i be a set for each $i \in I$. We denote this *I-indexed collection of sets* — a *dependent set* — by $X \colon I \to$ **Set** or more classically as $(X_i)_{i \in I}$. Choosing one element from one set in the collection would be denoted (i, x) where $x \in X_i$. Choosing an element from each set in the collection would give us a function $i \mapsto x_i$ where each $x_i \in X_i$. This is a sort of function we haven't seen before, at least in this form — its codomain *depends* on the element we are applying it to. Think of a vector field v: to each point p it assignes a tangent vector v_p in the tangent space *at* p. We can write the signature of such a function as

$$f : (i \in I) \to X_i.$$

We call this a *dependent function*, since its codomain depends on the element of its domain we are applying it to.

Definition 1.40 (Dependent sums and products of sets)**.** Let I be a set and $X \colon I \to$ **Set** be an I-indexed collection of sets. The sum $\sum_{i \in I} X_i$ (respectively, the *product* $\prod_{i \in I} X_i$)

[2] For those who clamor for polynomials in many variables, we will see in **??** that the multivariable story falls out of the one-variable story.

of this collection is the set

$$\sum_{i\in I} X_i = \{(i,x) \mid i \in I \text{ and } x \in X_i\} \qquad \text{and} \qquad \prod_{i\in I} X_i = \{f : (i \in I) \to X_i\}.$$

Example 1.41. If $I = 2 = \{1,2\}$ then a collection $X\colon I \to \mathbf{Set}$ is just two sets, say $X_1 = \{a,b,c\}$ and $X_2 = \{c,d\}$. The sum is the disjoint union

$$\sum_{i\in 2} X_i = X_1 + X_2 = \{(1,a),(1,b),(1,c),(2,c),(2,d)\}.$$

Its number of elements (cardinality) will always be the sum of the numbers of elements in X_1 and X_2. Similarly,

$$\prod_{i\in 2} X_i \cong X_1 \times X_2 = \{(a,c),(a,d),(b,c),(b,d),(c,c),(c,d)\}.$$

Exercise 1.42. Let I be a set and let $X_i = 1$ be a one-element set for each $i \in I$.

1. Show that there is an isomorphism of sets $I \cong \sum_{i\in I} 1$.
2. Show that there is an isomorphism of sets $1 \cong \prod_{i\in I} 1$.

As a special case, suppose $I := \varnothing$ and $X\colon \varnothing \to \mathbf{Set}$ is the unique empty collection of sets.

1. Is it true that $X_i = 1$ for each $i \in I$?
2. Show that there is an isomorphism of sets $0 \cong \sum_{i\in\varnothing} X_i$.
3. Show that there is an isomorphism of sets $1 \cong \prod_{i\in\varnothing} X_i$. ◊

Exercise 1.43. Let $X_{(-)} : I \to \mathbf{Set}$ be a set depending on an $i \in I$. There is a projection function $\pi_1 : \sum_{i\in I} X_i \to I$ defined by $\pi_1(i,x) = i$.

1. What is the signature of the second projection $\pi_2(i,x) = x$? (Hint: its a dependent function).
2. A *section* of a function $r : A \to B$ is a function $s : B \to A$ such that $r \circ s = \mathrm{id}_B$. Show that the dependent product is isomorphic to the set of sections of π_1:

$$\prod_{i\in I} X_i \cong \{s : I \to \sum_{i\in I} X_i \mid \pi_1 \circ s = \mathrm{id}_I\}.$$

◊

Where are we, and where are we going? We've define dependent sums and products of sets; that's where we are. Our goal is to define polynomial functors, e.g. $y^2 + 2y + 1$, and the maps between them. Since y^2, y, and 1 are functors, we just need to define sums of functors $\mathbf{Set} \to \mathbf{Set}$. But we might as well define products of functors at the same time, because they'll very much come in handy.

Dependent sums and products of functors Set → Set.

Definition 1.44 (Dependent sums and products of functors **Set** → **Set**)**.** For any two functors $F, G\colon \mathbf{Set} \to \mathbf{Set}$, let

$$(F+G)\colon \mathbf{Set} \to \mathbf{Set} \qquad \text{and} \qquad (F \times G)\colon \mathbf{Set} \to \mathbf{Set}$$

denote the functors that respectively assign to each $X \in \mathbf{Set}$ the sets

$$(F+G)(X) \coloneqq F(X) + G(X) \qquad \text{and} \qquad (F \times G)(X) \coloneqq F(X) \times G(X).$$

More generally, for any set I and functors $(F_i)_{i\in I}$, let

$$\sum_{i\in I} F_i \coloneqq \mathbf{Set} \to \mathbf{Set} \qquad \text{and} \qquad \prod_{i\in I} F_i \coloneqq \mathbf{Set} \to \mathbf{Set}$$

denote the functors that respectively assign to each $X \in \mathbf{Set}$ the sum and product of sets

$$\Big(\sum_{i\in I} F_i\Big)(X) \coloneqq \sum_{i\in I} F_i(X) \qquad \text{and} \qquad \Big(\prod_{i\in I} F_i\Big)(X) \coloneqq \prod_{i\in I} F_i(X).$$

We denote by $0, 1\colon \mathbf{Set} \to \mathbf{Set}$ the constant functors, which respectively assign 0 and 1 to each $X \in \mathbf{Set}$.

Proposition 1.45. Referring to the notation in Definition 1.44, the functors 0 and 1 are respectively an initial object and a terminal object in $\mathbf{Set}^{\mathbf{Set}}$, the operations $+$ and $\times$ are respectively a binary coproduct and a binary product in $\mathbf{Set}^{\mathbf{Set}}$, and the operations $\sum_{i\in I}$ and $\prod_{i\in I}$ are arbitrary coproducts and products in $\mathbf{Set}^{\mathbf{Set}}$.

Proof. By Example 1.41 and Exercise 1.42, it suffices to show that $\sum_{i\in I} F_i$ and $\prod_{i\in I} F_i$ are a sum and product in $\mathbf{Set}^{\mathbf{Set}}$. This itself is a special case of a more general fact, where sums and products are replaced by arbitrary colimits and limits, and where $\mathbf{Set}^{\mathbf{Set}}$ is replaced by an arbitrary functor category $\mathcal{C}^{\mathcal{D}}$, where $\mathcal{C}$ is a category that (like **Set**) has limits and colimits; see [**macLane1992sheaves**]. □

We've finally arrived: we can define the category of polynomial functors!

Poly, the category of polynomial functors

Definition 1.46 (Polynomial functors)**.** A *polynomial functor* is a functor $p\colon \mathbf{Set} \to \mathbf{Set}$ such that there exists a set I, sets $(p[i])_{i\in I}$, and an isomorphism

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

to a sum of representables. A *morphism of polynomial functors* is simply a natural transformation $p \to q$.

So (up to isomorphism), a polynomial functor is just a sum of representables.

Example 1.47. Consider the polynomial $y^2 + 2y + 1$. It denotes a functor **Set** $\to$ **Set**; what does this functor do to the set $X := \{a, b\}$? To be very precise and pedantic, let's say

$$I := 4 \qquad \text{and} \qquad p[1] := 2, \quad p[2] := 1, \quad p[3] := 1, \quad p[4] := 0$$

so that $p \cong \sum_{i \in I} y^{p[i]}$. Now we have

$$p(X) \cong \{(1, a, a), (1, a, b), (1, b, a), (1, b, b), (2, a), (2, b), (3, a), (3, b), (4)\}.$$

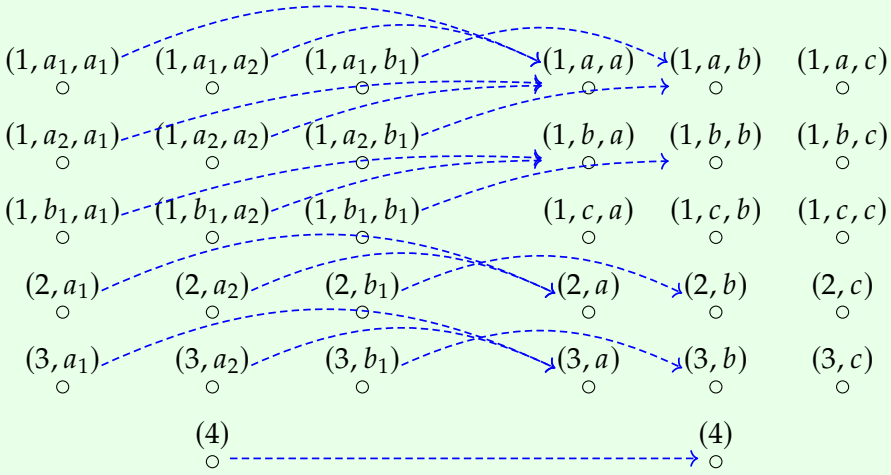
It has $(2^2 + 2 + 2 + 1)$-many, i.e. 9, elements. The representable summand y^A throws in all A-tuples from X, but it's indexed by the name of the summand. In particular if $A = \varnothing$ then it just records the empty tuple at that summand.

Exercise 1.48. In the pedantic style of Example 1.47, write out all the elements of $p(X)$ for p and X as follows:

1. $p := y^3$ and $X := \{4, 9\}$.
2. $p := 3y^2 + 1$ and $X := \{a\}$.
3. $p := 0$ and $X := \mathbb{N}$.
4. $p := 4$ and $X := \mathbb{N}$.
5. $p := y$ and $X := \mathbb{N}$. ◊

Example 1.49. Suppose $p := y^2 + 2y + 1$. As a functor **Set** $\to$ **Set** it should be able to act not only on sets but on functions. Let $X := \{a_1, a_2, b_1\}$, $Y := \{a, b, c\}$ and $f : X \to Y$ the function sending $a_1, a_2 \mapsto a$ and $b_1 \mapsto b$. The induced function $p(f) : p(X) \to p(Y)$

is shown below



Exercise 1.50. Let $p = y^2 + y$. Choose a function $f : 1 \to 2$ and write out the induced function $p(f) : p(1) \to p(2)$. ◊

Proposition 1.51. Let $p \coloneqq \sum_{i \in I} y^{p[i]}$ be an arbitrary polynomial functor. Then $I \cong p(1)$, so there is an isomorphism of functors

$$p \cong \sum_{i \in p(1)} y^{p[i]}. \tag{1.52}$$

Proof. We need to show that $I \cong p(1)$; the latter claim follows directly. In Exercise 1.42 it was shown that $I \cong \sum_{i \in I} 1$, so we just need to show that $(y^A)(1) \cong 1$ for any $A \in \mathbf{Set}$. But $1^A \cong 1$ because there is a unique function $A \to 1$ for any A. □

The notation in (1.52) will be how we denote arbitrary polynomials from now on.

Exercise 1.53. We saw in Proposition 1.51 that for any polynomial p, e.g. $p = y^3 + 3y^2 + 4$, the set $p(1)$ gives back the set of summands, in this case 8.

What does $p(0)$ give you? ◊

1.2.3 Morphisms between polynomial functors are dependent lenses

For a seasoned category theorist, we've already said enough to calculate the set $\mathbf{Poly}(p, q)$ of morphisms between polynomials p and q: it's the set of natural transformations between p and q, which are sums of representable functors. But it is worth going over this subject at a more leisurely pace, because it helps us insert some intuition that will be very useful later.

In particular, we're going to think about polynomials like p and q using the intuition from the beginning of the chapter, around the table in (1.3). We said that we'd think of p as a arena: a set of decisions you can make and what options you have. That is, $p(1)$ is conceived as a set of decisions, and for every decision $i \in p(1)$, we conceive of the set $p[i]$ as the options or *options* available in that decision. Here's how we write p as an arena:

$$p(1) \xrightarrow{p[-]} \mathbf{Set}$$

Again, every decision $i \in p(1)$ gets a set $p[i]$ of options: that's the whole content of p.

Definition 1.54. An *arena* for the dependent doctrine is a *dependent set* $A^-_{(-)} : A^+ \to \mathbf{Set}$; that is, functor from the discrete category on a set A^+ into the category of sets. the set A^-_a that each direction a^- lives in depends on the position $a \in A^+$. We will write the arena $A^- : A^+ \to \mathbf{Set}$ as

$$\begin{pmatrix} A^-_a \\ a \in A^+ \end{pmatrix}.$$

We have just noted that every polynomial p gives an arena $\left(\begin{smallmatrix} p[i] \\ i \in p(1) \end{smallmatrix}\right)$ whose positions are the decisions in the polynomial and whose directions are the options.

The maps $f : p \to q$ of polynomials are then the ways to *delegate* p's decisions to q. Every one of p's decisions, say $i \in p(1)$, is passed forward to a decision $j \in q(1)$ for q to make, and every choice $d \in q[j]$ that q could make among its options is passed back as some choice $c \in p[i]$ among p's options.

```
record Delegation (w : Arena) (w' : Arena) : Set where
  field
    passfwd : (decn p) -> (decn q)
    passbck : (d : decn p) -> posy q (passfwd d) -> posy p d
```

This structure of passing information forward and backwards should look familiar; its a sort of lens! The difference between this sort of lens and the sort that we saw in earlier chapters is that the *signature* of the passback can depend on which decision one is using. We will therefore refer to these lenses as *dependent lenses*.

Definition 1.55. A *dependent lens* $\left(\begin{smallmatrix} f^\sharp \\ f \end{smallmatrix}\right) : \left(\begin{smallmatrix} A^-_a \\ a \in A^+ \end{smallmatrix}\right) \leftrightarrows \left(\begin{smallmatrix} B^-_b \\ b \in B^+ \end{smallmatrix}\right)$ consists of

- a *passforward* function $f^+ : A^+ \to B^+$, and
- a dependent *passback* fucntion $f^\sharp : (a \in A^+) \to (B^-_{f(a)} \to A^-_a)$. That is, for every $a \in A^+$, we have a passback function $f^\sharp(a, -) : B^-_{f(a)} \to A^-_a$.

The composite of the dependent lens $\left(\begin{smallmatrix} f^\sharp \\ f \end{smallmatrix}\right) : \left(\begin{smallmatrix} A^-_a \\ a \in A^+ \end{smallmatrix}\right) \leftrightarrows \left(\begin{smallmatrix} B^-_b \\ b \in B^+ \end{smallmatrix}\right)$ with the dependent

lens $\begin{pmatrix} f^\sharp \\ f \end{pmatrix} : \begin{pmatrix} B_b^- \\ b \in B^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} C_c^- \\ c \in C^+ \end{pmatrix}$ is given by

$$\begin{pmatrix} g^\sharp \\ g \end{pmatrix} \circ \begin{pmatrix} f^\sharp \\ f \end{pmatrix} := \begin{pmatrix} (a, c^-) \mapsto f^\sharp(a, g^\sharp(f(a), c^-)) \\ g \circ f \end{pmatrix}.$$

This gives us the category of *dependent lenses* **DLens**.

Proposition 1.56. Consider the functor $\mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ sending each set A to the slice category $\mathbf{Set}/A$, and sending each $f : B \to A$ to the functor $f^* : \mathbf{Set}/A \to \mathbf{Set}/B$. Then **DLens** is equivalent to the category *LSpiz: notation?* of lenses for this doctrine.

Proof. We define a functor $F : \mathbf{DLens} \to L$ as follows. An arena $A := \begin{pmatrix} A_a^- \\ a \in A^+ \end{pmatrix} \in \mathrm{Ob}\,\mathbf{DLens}$, assigns a set A_a^- to each $a \in A^+$; let $X := \sum_{a \in A^+} A_a^-$ with its obvious map $X \to A^+$. Then $F(A) := \begin{pmatrix} X \\ A \end{pmatrix}$ is notation for an object in L. We leave it to the reader to give the functor on morphisms, and show that it is fully faithful and essentially surjective. □

Exercise 1.57. Complete the proof of Proposition 1.56 as follows.

1. To what morphism in L does F assign the morphism $\begin{pmatrix} f^\sharp \\ f \end{pmatrix} : \begin{pmatrix} A_a^- \\ a \in A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B_b^- \\ b \in B^+ \end{pmatrix}$?
2. Show that F is fully faithful.
3. Show that F is essentially surjective. ◊

Remark 1.58. Let $p := \sum_{i \in p(1)} y^{p[i]}$ and $q := \sum_{j \in q(1)} y^{q[j]}$ be polynomials. We can think of a dependent lens $p \to q$ as in Definition 1.55 diagrammatically as follows:

$$\begin{array}{ccc} p(1) & \xrightarrow{f_1} & q(1) \\ & \overset{f^\sharp}{\Leftarrow} & \\ p[-] \searrow & & \swarrow q[-] \\ & \mathbf{Set} & \end{array} \tag{1.59}$$

That is, f_1 is a function (or functor between discrete categories) and $f^\sharp$ is a natural transformation: for each $i \in p(1)$ with $j := f_1(i)$, there is a function $f_i^\sharp : q[j] \to p[i]$.

Dependent lenses are indeed a generalization of the lenses we saw in **??**.

Proposition 1.60. There is a fully faithful inclusion $\mathbf{Lens}_{\mathbf{Set}} \hookrightarrow \mathbf{DLens}$ interpreting the coKleisli arena $\begin{pmatrix} A^- \\ A^+ \end{pmatrix}$ as the *constant* dependent arena $\begin{pmatrix} A^- \\ a \in A^+ \end{pmatrix}$ (that is, the constant functor $a \mapsto A^- : A^+ \to \mathbf{Set}$).

Proof. We just need to check that the definition of dependent lens specializes appropriately to the definition of lens. So, suppose we have constant dependent arenas $\begin{pmatrix} A^- \\ a \in A^+ \end{pmatrix}$ and $\begin{pmatrix} B^- \\ b \in B^+ \end{pmatrix}$. Then a dependent lens $\begin{pmatrix} f^\sharp \\ f \end{pmatrix}$: □

Lemma 1.61. Using $\sum$s and $\prod$s, we can give a slick definition of dependent lens. We have a bijection

$$\mathbf{DLens}\left(\begin{pmatrix} A_a^- \\ a \in A^+ \end{pmatrix}, \begin{pmatrix} B_b^- \\ b \in B^+ \end{pmatrix}\right) \cong \prod_{a \in A^+} \sum_{b \in B^+} (A_a^-)^{(B_b^-)}$$

Proof. This is a special case of the upcoming Proposition 1.74, so we'll just sketch the proof here.

A dependent lens $\begin{pmatrix} f^\sharp \\ f \end{pmatrix} : \begin{pmatrix} A_a^- \\ a \in A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B_b^- \\ b \in B^+ \end{pmatrix}$ consists of a map $f : A^+ \to B^+$ and a depedent map $f^\sharp : (a \in A^+) \to (B^-_{f(a)} \to A^-_a)$, while an element of the right ahnd side is dependent function

$$(f, f^\sharp) : (a \in A^+) \to \sum_{b \in B^+} (B_b^- \to A_a^-)$$

which takes an element of a and yields a pair $(f(a), f^\sharp(a, -))$. These are just two different ways of pairing up the data. □

It seems we are observing that the category of polynomial functors is equivalent to the category of dependent lenses! Let's state this mathematically and prove it.

Proposition 1.62. Taking a polynomial p to its arena $\begin{pmatrix} p[i] \\ i \in p(1) \end{pmatrix}$ gives an equivalence between **Poly** and the category of dependent lenses.

To summarize, we have an isomorphism

$$\mathbf{Poly}(p, q) \cong \prod_{i \in p(1)} \sum_{j \in q(1)} p[i]^{q[j]} \cong \mathbf{DLens}\left(\begin{pmatrix} p[i] \\ i \in p(1) \end{pmatrix}, \begin{pmatrix} q[j] \\ j \in q(1) \end{pmatrix}\right) \tag{1.63}$$

and composition of polynomial morphisms corresponds to composition of dependent lenses.

Proof. We have an assignment $p \mapsto \begin{pmatrix} p[i] \\ i \in p(1) \end{pmatrix}$ which sends an object of **Poly** — a polynomial functor — to an object of the category of dependent lenses — an arena. We want to show that this object assignment gives an equivalence of categories. First, we can note that it is surjective up to isomorphism on objects, since to every arena $\begin{pmatrix} A_a^- \\ a \in A^+ \end{pmatrix}$ we get the polynomial functor

$$\sum_{a \in A^+} y^{A_a^-}$$

and if we take the arena corresponding to the polynomial, we back where we started since $\sum_{a \in A^+} 1^{A_a^-} \cong A^+$.

Next, we'll show that the set of natural transformations $\mathbf{Poly}(p, q)$ between polynomial functors p and q is isomorphic to the set of dependent lenses between their arenas.

We have the following isomorphisms

$$\begin{aligned}
\mathbf{Poly}(p,q) &= \mathsf{Nat}\Big(\sum_{i\in p(1)} y^{p[i]}, \sum_{j\in q(1)} y^{q[j]}\Big) && \text{Definitions}\\
&\cong \prod_{i\in p(1)} \mathsf{Nat}\Big(y^{p[i]}, \sum_{j\in q(1)} y^{q[j]}\Big) && \text{Univ. prop. of coproduct}\\
&\cong \prod_{i\in p(1)} \sum_{j\in q(1)} p[i]^{q[j]} && \text{Yoneda lemma}
\end{aligned}$$

This proves the first part of we called the summary. It says that a morphism $p \to q$ consists of: for every $i \in p(1)$ both a choice of $j \in q(1)$ and a function from $q[j] \to p[i]$; the first part determines f (i.e. $f(i) = j$) and the second part just gives $f^\sharp_i$ directly for each i.

We can now note that by Lemma 1.61, $\prod_{i\in p(1)} \sum_{j\in q(1)} p[i]^{q[j]}$ is isomorphic to the set of dependent lenses $\left(\begin{smallmatrix} p[i] \\ i\in p(1)\end{smallmatrix}\right) \leftrightarrows \left(\begin{smallmatrix} q[j] \\ j\in q(1)\end{smallmatrix}\right)$.

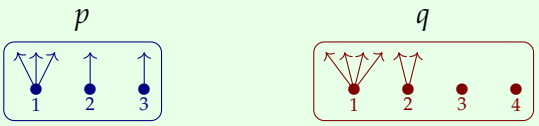
Finally, it remains to show that these isomorphisms are functorial. We leave this to Exercise 1.64 □

Exercise 1.64. Complete the proof of Proposition 1.62. ◊

Remark 1.65. The fact in (1.63) that a natural transformation between polynomial functors is equivalently a dependent lens will be quite important to us; we will call it our *main formula* for the set of polynomial maps.

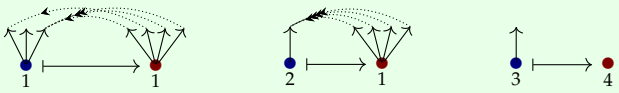
In fact, this formula will be so important to us that we will often blur the difference between a dependent lens $\left(\begin{smallmatrix} f^\sharp \\ f\end{smallmatrix}\right) : \left(\begin{smallmatrix} p[i] \\ i\in p(1)\end{smallmatrix}\right) \leftrightarrows \left(\begin{smallmatrix} q[j] \\ j\in q(1)\end{smallmatrix}\right)$ with the natural transformation $p \to q$ it corresponds to.

Example 1.66. Let $p := y^3 + 2y$ and $q := y^4 + y^2 + 2$



To give a dependent lens $p \to q$, one delegates each p-decision $i \in p(1)$ to a q-decision $j \in q(1)$, and then each option in $q[j]$ is passed back to one in $p[i]$.

How many ways are there to do this? Before answering this, let's just pick one.



This represents one morphism $p \to q$.

So how many different morphisms are there from p to q? One way to figure it out is to use the formula (1.63), but that might need some explaining; we go through such notation in an interlude after Exercise 1.67. For now, we can just count "pre-theoretically".

The first p-decision can be delegated to q-decision 1, 2, 3, or 4. Delegating it to the first q-decision requires choosing how each of the four options ($q[1] = 4$) are to be assigned one of $p[1] = 3$ options; there are 3^4 ways to do this. Similarly, we can calculate all the ways to delegate the first p-decision: there are $3^4 + 3^2 + 3^0 + 3^0 = 92$.

The second p-decision can be delegated to q-decision 1, 2, 3, or 4. There are $1^4 + 1^2 + 1^0 + 1^0 = 4$ ways to do this; similarly there are four ways to delegate the third p-decision to a q-decision.

In total, there are $92 * 4 * 4 = 1472$ morphisms $p \to q$, just twenty less than the number of years between Jesus's famous year and Columbus's famous year. Coincidence?

Exercise 1.67.

1. Draw the poly's (unions of corollas as in Eq. (1.2)) associated to $p \coloneqq y^3 + y + 1$, $q \coloneqq y^2 + y^2 + 2$, and $r \coloneqq y^3$.
2. Give an example of a morphism $p \to q$ and draw it as we did in Example 1.66.
3. Explain your morphism as a "delegation" in the sense of Section 1.2.3.
4. Explain in those terms why there can't be any morphisms $p \to r$. ◊

Exercise 1.68. For any polynomial p and set A, e.g. $A = 2$, the Yoneda lemma gives an isomorphism $p(A) \cong \mathbf{Poly}(y^A, p)$.

1. Choose a polynomial p and draw the associated poly as well as y^2.
2. Count all the maps $y^2 \to p$.
3. Is it equal to $p(2)$? ◊

Example 1.69 (Derivatives). We won't have much opportunity to use the following, but it may be interesting. The *derivative* of a polynomial p, denoted $\dot{p}$, is defined as follows:

$$\dot{p} \coloneqq \sum_{i \in p(1)} \sum_{d \in p[i]} y^{p[i] - \{d\}}.$$

For example, if $p \coloneqq y^{\{U,V,W\}} + \{A, B\} y^{\{X\}}$ then

$$\dot{p} = \{U\} y^{\{V,W\}} + \{V\} y^{\{U,W\}} + \{W\} y^{\{U,V\}} + \{(A, X), (B, X)\} y^0.$$

Up to isomorphism $p \cong y^3 + 2y$ and $\dot{p} \cong 3y^2 + 2$.

A morphism $f : p \to \dot{q}$ can be interpreted in terms of delegations from p to q such that each position in p explicitly selects a q-option to be avoided. More precisely, for

each $i \in p(1)$ we have $f_1(i) = (j, d) \in \sum_{j \in q(1)} q[j]$, i.e. a choice of q-position j, as usual, together with a chosen option $d \in q[j]$. Then every q-option *other than* d is passed back to an option in $p[i]$.

Exercise 1.70. The derivative is not very well-behaved category-theoretically. However, it is intriguing.

1. Is there always a canonical map $p \to \dot{p}$?
2. Is there always a canonical map $\dot{p} \to p$?
3. If given a map $p \to q$, does one get a map $\dot{p} \to \dot{q}$?
4. What is the relationship between $\dot{p}(1)$ and the total number of leaves in p?
5. In terms of problems and solutions, explain what a map $p \to \dot{q}$ would say.
6. Read the formula for $\otimes$ and $[-,-]$ in (1.88) and (1.158) and find a formula for a map $p \otimes [p, y] \to p'$ that works for any $p \in$ **Poly**.
7. When talking to someone who explains maps $p \to \dot{q}$ in terms of "avoiding options", how might you describe what is modeled by a map $py \to q$?

◊

Working with $\sum$ and $\prod$ formulas. Here's how to understand and work with a set that involves $\sum$'s and $\prod$'s, e.g. something like

$$\sum_{i \in I} \prod_{j \in J(i)} \sum_{k \in K(i,j)} X(i, j, k). \tag{1.71}$$

To understand it, you ask "what is involved in choosing an element of this set?"

The answer given by the formula (1.71) begins by saying "To choose an element of (1.71), first..." and then you begin pronouncing the formula. Every time you see $\sum_{i \in I} \ldots$, say "choose an element $i \in I$; then, ...". Every time you see $\prod_{j \in J}$, say "for each $j \in J$, ...". At the very end when you see some X, say "conclude by choosing an element of X".

So the formula in (1.71) would be pronounced as follows:

> To choose an element of (1.71), first choose an element $i \in I$. Then, for every $j \in J(i)$, choose an element $k \in K(i, j)$; then, conclude by choosing an element of $X(i, j, k)$.

Note that the choice of $k \in k(i, j)$ can depend on i and j; it must be able to because these sets $K(i, j)$ can be different for different i, j.

Example 1.72 (Notation for $\sum \prod$ stuff). Here we give notation for the elements of a set involving $\sum$'s and $\prod$'s such as that in (1.71).

Let $I = \{1, 2\}$, let $J(1) = \{j\}$ and $J(2) := \{j, j'\}$, let $K(1, j) := \{k_1, k_2\}$, $K(2, j) := \{k_1\}$,

and $K(2, j') := \{k'\}$, and let $X(i, j, k) = \{x, y\}$ for all i, j, k. Now the formula

$$\sum_{i \in I} \prod_{j \in J(i)} \sum_{k \in K(i,j)} X(i, j, k)$$

has been given meaning as an actual set. Here is a list of all eight of its elements:

$$\left\{ \begin{array}{cc} \big(1, j \mapsto (k_1, x)\big), & \big(1, j \mapsto (k_1, y)\big), \\ \big(2, j \mapsto (k_1, x), j' \mapsto (k_1, x)\big), & \big(2, j \mapsto (k_1, x), j' \mapsto (k_1, x)\big), \\ \big(2, j \mapsto (k_1, x), j' \mapsto (k_1, x)\big), & \big(2, j \mapsto (k_1, x), j' \mapsto (k_1, x)\big) \end{array} \right. \quad \left. \begin{array}{c} \big(1, j \mapsto (k_2, x)\big), \quad \big(1, j \mapsto (k_2, y)\big), \\ \\ \end{array} \right\}$$

In each case, we first chose an element $i \in I$, either 1 or 2. Then for each $j \in J(i)$ we chose an element $k \in K(i, j)$; then we concluded by choosing an element of $X(i, j, k)$.

Exercise 1.73. With I, J, K, and X as in Example 1.72, consider the set

$$\prod_{i \in I} \sum_{j \in J(i)} \prod_{k \in K(i,j)} X(i, j, k).$$

1. Pronounce it using the "To choose an element" formulation just below (1.71).
2. How many elements are in it?
3. Write out three of them in the style of Example 1.72. ◊

The following is sometimes called the *type-theoretic axiom of choice* or the *completely distributive property*, in this case of **Set**. It is almost trivial, once you understand what it's saying; for example once the statement is written in agda, its proof is one short line of agda code.

Proposition 1.74 (Pushing $\prod$ past $\sum$)**.** For any set I, collection of sets $\{J(i)\}_{i \in I}$, and collection of sets $\{X(i, j)\}_{i \in I, j \in J(i)}$, we have a bijection

$$\sum_{j \in \prod_{i \in I} J(i)} \prod_{i \in I} X(i, j(i)) \cong \prod_{i \in I} \sum_{j \in J(i)} X(i, j). \tag{1.75}$$

Proof. We'll do this the old-fashioned way: by giving a map from left to right, and a map from right to left, and showing that they are mutually inverse.

First, let's go from left to right. An element of the set on the left is a pair of a function $j \in \prod_{i \in I} J(i)$ together with a function $f \in \prod_{i \in I} X(i, j(i))$. We then get an element of the set on the right as follows:

$$i \mapsto (j(i), f(i)).$$

Now, let's go right to left. An element of the set on the right is a function $f \in \prod_{i \in I} \sum_{j \in J(i)} X(i, j)$. We can then form the following pair in the left hand set:

$$((i \mapsto \pi_1 f(i)), (i \mapsto \pi_2 f(i)))$$

where $\pi_1 f(i)$ is the first component of the pair $f(i)$ and $\pi_2 f(i)$ is the second component.

Now, we just need to check that a round trip takes us back where we were. If we start on the left, our round trip gives us the pair

$$((i \mapsto \pi_1(j(i), f(i))), (i \mapsto \pi_2(j(i), f(i)))).$$

But $\pi_1(j(i), f(i)) = j(i)$ and $\pi_2(j(i), f(i)) = f(i)$ by definition, so we're back where we started. On the other hand, starting on the right gives us the function

$$i \mapsto (\pi_1 f(i), \pi_2 f(i)).$$

But again, since $f(i)$ is the pairing of its components $\pi_1(f(i))$ and $\pi_2(f(i))$, we're back where we started.

□

Exercise 1.76. Use Proposition 1.74 to give another proof of Lemma 1.61. (Hint: express the left hand side as a dependent sum.) ◊

Below, e.g. in Exercise 1.77, you'll often see alternating sums and products; using Eq. (1.75), you can always put it into sums-then-products ("disjunctive normal") form.

When $J(i) = J$ does not depend on $i \in I$, the formula in (1.75) is much easier:

$$\prod_{i \in I} \sum_{j \in J} X(i,j) \cong \sum_{j : I \to J} \prod_{i \in I} X(i,j).$$

Exercise 1.77. Let p, q be polynomials as in Proposition 1.62. Recall from our main formula (1.63) that there is an isomorphism between polynomia$\mathbf{Poly}(p,q) \cong \prod_{i \in p(1)} \sum_{j \in q(1)} p[i]^{q[j]}$.

1. Is it true that the following are isomorphic?

$$\mathbf{Poly}(p,q) \cong^? \prod_{i \in p(1)} \sum_{j \in q(1)} \prod_{d \in q[j]} \sum_{c \in p[i]} 1 \tag{1.78}$$

2. Is it true that there is an isomorphism

$$\mathbf{Poly}(p,q) \cong^? \sum_{f_1 : p(1) \to q(1)} \prod_{j \in q(1)} \mathbf{Set}\Big(q[j], \prod_{\{i \in p(1) | f_1(i) = j\}} p[i]\Big). \tag{1.79}$$

3. If the answer to #2 is "yes", then say how any element of the right-hand side gives a way of delegating decisions from p to q. If "no", give intuition for why the two sets are not isomorphic. ◊

Back to maps of polynomials After that interlude, you are hopefully more comfortable with our main formula Eq. (1.63) describing the set of morphisms between two arbitrary polynomial functors as dependent lenss. Let's practice.

Example 1.80 (Constants are sent to constants). Any natural transformation $p \to q$ must send constants in p to constants in q. So if p has constant terms and q doesn't, e.g. if $p = y^2 + 1$ and $q = y^3$, then there are no maps $p \to q$.

Indeed, let's more generally say $p \coloneqq \sum_{i \in p(1)} y^{p[i]}$ and $q \coloneqq \sum_{j \in q(1)} y^{q[j]}$ are polynomials, and that we have a map $p \to q$ corresponding to a dependent lens $\left(\begin{smallmatrix} p[i] \\ i \in p(1) \end{smallmatrix}\right) \leftrightarrows \left(\begin{smallmatrix} q[j] \\ j \in q(1) \end{smallmatrix}\right)$. If there is some $i_0 \in p(1)$ such that $p[i_0] = 0$, then the formula says that for i_0 our map has chosen a $j \in q(1)$ and a function $q[j] \to p[i]$. But in order for such a function to exist, if $p[i]$ is empty, $q[j]$ must be too.

Exercise 1.81. For each of the following p, q, count the number of natural transformations $p \to q$:

1. $p = y^3, \quad q = y^4$.
2. $p = y^3 + 1, \quad q = y^4$.
3. $p = y^3 + 1, \quad q = y^4 + 1$.
4. $p = 4y^3 + 3y^2 + y, \quad q = y$.
5. $p = 4y^3, \quad q = 3y$. ◊

Exercise 1.82 (A functor **Top** $\to$ **Poly**). This exercise is for those who know topological spaces and the maps between them. It will not be used again in this book.

1. Suppose that X is a topological space. Organize its points and their neighborhoods into a polynomial p_X.
2. Give a formula by which any continuous map $X \to Y$ induces a map of polynomials $p_X \to p_Y$.
3. Show that your formula defines a functor.
4. Is it full? Faithful? ◊

1.2.4 Prepare for dynamics

As beautiful as the category **Poly** is—and to be clear we have not really begun to say what is so special about it—discussing its virtues is not our goal. We want to use it!

Polynomial functors are a setting in which to speak about dynamics, data, and decision. We want the reader to understand this deeply as they go through the mathematics. So in order to make the story a bit more seamless, we discuss a few relevant aspects of **Poly** that we can use immediately.

Finite products The category **Poly** has limits and colimits, is Cartesian closed, has epi-mono factorizations, is completely distributive, etc., etc. However, in order to tell a good story of dynamics, we only need finite products right now. These will be useful for letting many different interfaces control the same internal dynamics.

Proposition 1.83. The category **Poly** has finite products.

Proof. We will see that 1 is a terminal object and that the product of p and q in **Poly** is the usual product of p and q as polynomials. That is, if $p \coloneqq \sum_{i \in p(1)} y^{p[i]}$ and $q \coloneqq \sum_{j \in q(1)} y^{q[j]}$ are in standard notation, then

$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i]+q[j]}. \tag{1.84}$$

We leave the proof as an exercise; see Exercise 1.85. □

Exercise 1.85. Use Eq. (1.63) to prove Proposition 1.83 using dependent lenses. In particular:

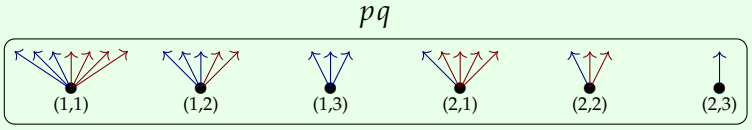
1. Prove that 1 is terminal in **Poly**.
2. Prove that for polynomials p, q, their product is given by the formula in (1.84). ◊

We will begin writing pq rather than $p \times q$:

$$pq \coloneqq p \times q \tag{Notation}$$

Example 1.86. We can draw the product of two polynomials in terms of their associated poly's. Let $p \coloneqq y^3 + y$ and $q \coloneqq y^4 + y^2 + 1$.

Then $pq \cong y^7 + 2y^5 + 2y^3 + y$. In pictures, we take all pairs of decisions, and for each pair we take the union the options.



Parallel product There is a closely related monoidal structure on **Poly** that will be useful for putting dynamical systems in parallel and then wiring them together.

Definition 1.87. Let p and q be polynomials. When they are expressed in standard notation, their *parallel product*, denoted $p \otimes q$ is given by the formula:

$$p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i] \times q[j]}. \tag{1.88}$$

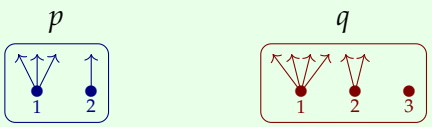
On arenas, this is defined by

$$\begin{pmatrix} p[i] \\ i \in p(1) \end{pmatrix} \otimes \begin{pmatrix} q[j] \\ j \in q(1) \end{pmatrix} := \begin{pmatrix} p[i] \times q[j] \\ (i,j) \in (p \times q)(1) \end{pmatrix}. \tag{1.89}$$

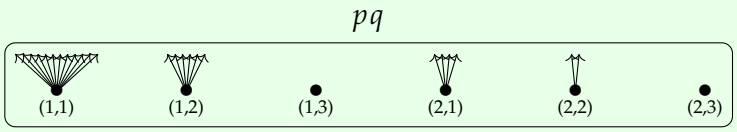
One should compare this with the formula for product of polynomials shown in (1.84). The difference is that parallel product multiplies exponents where regular product adds them.

Remark 1.90. The reason we call this the parallel product is because it generalizes the parallel product for lenses in **??**.

Example 1.91. We can draw the product of two polynomials in terms of their associated poly's. Let $p := y^3 + y$ and $q := y^4 + y^2 + 1$.



Then $p \otimes q \cong y^{12} + y^6 + y^4 + y^2 + 2$. In pictures, we take all pairs of decisions, and for each pair we take the product of the options.



Exercise 1.92. What is the parallel product of monomials

$$A_1 y^{B_1} \otimes A_2 y^{B_2}?$$

◊

Exercise 1.93. Let $p := y^3 + y$ and $q := 2y^4$. Draw the following arenas using corollas:

1. Draw p and q as arenas.

2. Draw $pq = p \times q$ as an arena.
3. Draw $p \otimes q$ as an arena. ◊

Exercise 1.94. Consider the polynomials $p \coloneqq 2y^2 + 3y$ and $q \coloneqq y^4 + 3y^3$.

1. What is $p \times q$?
2. What is $p \otimes q$?
3. What is the product of the following purely formal expression we'll see *only this once!*:
$$(2{\cdot}2^y + 3{\cdot}1^y + 1) \cdot (1{\cdot}4^y + 3{\cdot}3^y + 2)$$
These are called Dirichlet series.
4. Do you see a connection between what the last two parts? If so, would you say that it justifies the name "Dirichlet product" as another name for the parallel product $\otimes$?

◊

Exercise 1.95. What is $(3y^5 + 6y^2) \otimes 4$? Hint: $4 = 4y^0$. ◊

Exercise 1.96. Let $p, q, r \in$ **Poly** be any polynomials.

1. Show that there is an isomorphism $p \otimes y \cong p$.
2. Show that there is an isomorphism $(p \otimes q) \otimes r \cong p \otimes (q \otimes r)$.
3. Show that there is an isomorphism $(p \otimes q) \cong (q \otimes p)$. ◊

In Exercise 1.96 we have gone most of the way to proving that (**Poly**, $\otimes$, y) is a symmetric monoidal category.

Proposition 1.97. The category **Poly** has a symmetric monoidal structure $(y, \otimes)$ where $\otimes$ is the parellel product from Definition 1.87.

Sketch of proof. Given dependent lenses $f\colon p \to p'$ and $g\colon q \to q'$ we need to give a dependent lens $(f \otimes g)\colon (p \otimes q) \to (p' \otimes q')$. On positions, define

$$(f \otimes g)_1(i, j) \coloneqq \big(f_1(i), g_1(j)\big)$$

On directions at $(i, j) \in p(1) \times q(1)$, define

$$(f \otimes g)^\sharp_{(i,j)}(d, e) \coloneqq \big(f_i^\sharp(d), g_j^\sharp(e)\big).$$

We have not proven all the coherences between the various isomorphisms, but we ask the reader to take it on trust or to check it for themselves. □

Exercise 1.98.

1. If $p = A$ and $q = B$ are constant polynomials, what is $p \otimes q$?
2. If $p = A$ is constant and q is arbitrary, what can you say about $p \otimes q$?
3. If $p = Ay$ and $q = By$ are linear polynomials, what is $p \otimes q$?
4. For arbitrary $p, q \in \mathbf{Poly}$, what is the relationship between the sets $(p \otimes q)(1)$ and $p(1) \times q(1)$? ◊

Exercise 1.99. Which of the following classes of polynomials are closed under $\otimes$? Note also whether they contain y.

1. The set $\{Ay^0 \mid A \in \mathbf{Set}\}$ of constant polynomials.
2. The set $\{Ay \mid A \in \mathbf{Set}\}$ of linear polynomials.
3. The set $\{Ay + B \mid A, B \in \mathbf{Set}\}$ of affine polynomials.
4. The set $\{Ay^2 + By + C \mid A, B, C \in \mathbf{Set}\}$ of quadradic polynomials.
5. The set $\{Ay^B \mid A, B \in \mathbf{Set}\}$ of monomials.
6. The set $\{Sy^S \mid S \in \mathbf{Set}\}$ of systematic polynomials.
7. The set $\{p \in \mathbf{Poly} \mid p(1) \text{ is finite }\}$.

What is the smallest class of polynomials that's closed under $\otimes$ and contains y? ◊

Exercise 1.100. Show that for any $p_1, p_2, q \in \mathbf{Poly}$ there is an isomorphism

$$(p_1 + p_2) \otimes q \cong (p_1 \otimes q) + (p_2 \otimes q)$$ ◊

Proposition 1.101. For any monoidal structure $(I, \odot)$ on **Set**, there is a corresponding monoidal structure on **Poly** with unit y^I. It is given by Day convolution, and it distributes over $+$.

In the case of $(0, +)$ and $(1, \times)$, this procedure returns the $(1, \times)$ and $(y, \otimes)$ monoidal structures respectively.

Proof. Day convolution always takes a monoidal structure on **Set** and returns one on functors **Set** $\to$ **Set**; the issue is whether **Poly** is closed under the resulting monoidal product. We will show that it is, roughly because every polynomial is a coproduct of representables.

** □

Exercise 1.102.

1. Show that the operation $(A, B) \mapsto A + AB + B$ on **Set** is associative.
2. Show that 0 is unital for the above operation.

3. Let $(1, \odot)$ denote the corresponding monoidal product on **Poly**. What is $(y^3 + y) \odot (2y^2 + 2)$? ◊

⊗-monoids in Poly

Definition 1.103. A *⊗-monoid in **Poly*** consists of a polynomial m, a map $(*)\colon m \otimes m \to m$, and a position $e\colon y \to m$ that form a monoid in $(\mathbf{Poly}, y, \otimes)$.

These have a kind of swarm-like semantics, making individuals able to summarize the positions of a group of subordinates, as well as distribute instructions to those subordinates in a coherent way.

Exercise 1.104. Explain how $\odot$ has the "swarm-like" semantics described above. ◊

Example 1.105 (Monoids in **Set** give linear ⊗-monoids). If $(M, e, (*))$ is a monoid in $(\mathbf{Set}, 1, \times)$ then the linear polynomial My carries a corresponding ⊗-monoid, roughly because $Ay \otimes By \cong (A \times B)y$. This construction is functorial and fully faithful: a map between monoids in **Set** can be identified with a map between ⊗-monoids.

Example 1.106 (Comonoids in **Set** give representable ⊗-monoids). Every comonoid (S, ϵ, δ) in $(\mathbf{Set}, 1, \times)$ gives rise to a ⊗-monoid structure on y^S. This construction is again (contravariantly) functorial and fully faithful.

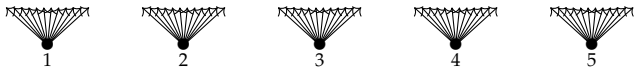
Example 1.107. Let $m \coloneqq \mathbb{R}_{\geq 0} y^{\mathbb{R}_{\geq 0}}$. Take $e = 0$ and $(a, b) \mapsto (a + b, t \mapsto (\frac{at}{a+b}, \frac{bt}{a+b}))$. This forms a ⊗-monoid.

Proposition 1.108. If m, n are the carriers of ⊗-monoids, then $m \otimes n$ naturally also carries a ⊗-monoid structure.

Proposition 1.109 (Free ⊗-monoid)**.** The forgetful functor from ⊗-monoids to **Poly** has a left adjoint.

Proof. The carrier p' of the free ⊗-monoid on p has positions $p'(1) \cong \mathsf{List}(p(1))$, i.e. lists of positions in p. Given a position $\ell = (i_1, \ldots, i_n)$, the direction-set there is given by $\prod_{j \in \mathsf{n}} p[j]$. ** □

Bimorphic lenses Monomials are special polynomials: those of the form Ay^B for sets A, B. Here's a picture of $5y^{12}$:



The formula for morphisms between these is particularly simple:

$$\mathbf{Poly}\left(A_1y^{B_1},\, A_2y^{B_2}\right) \cong \mathbf{Set}(A_1, A_2) \times \mathbf{Set}(A_1 \times B_2, B_1)$$

It says that to give a morphism from one monomial to another, you just need to give two functions. Let's rewrite it to make those two functions explicit:

$$\mathbf{Poly}\left(A_1y^{B_1},\, A_2y^{B_2}\right) \cong \left\{ (f_1, f^\sharp) \;\middle|\; \begin{array}{l} f_1 \colon A_1 \to A_2 \\ f^\sharp \colon A_1 \times B_2 \to B_1 \end{array} \right\}$$

Ordinarily, the $f^\sharp$ is more involved in that its type depends on f_1, but for monomials every decision has the same number of options, so to speak.

The monomials in **Poly** and the morphisms between them forms a full subcategory of **Poly**, and it has been called the *the category of bimorphic lenses* [**hedges2018limits**]. It comes up in functional programming. The morphisms $A_1y^{B_1} \to A_2y^{B_2}$ break up into two functions which people give special names:

$$\begin{aligned} &\text{get} \colon A_1 \to A_2 \\ &\text{set} \colon A_1 \times B_2 \to B_1 \end{aligned} \tag{1.110}$$

The idea is that each A-decision "gets" a B-decision, and every option there in B "sets" an option back in A

We will use the term *lens* when we want to remind the reader that morphisms between monomials are much simpler than those between arbitrary polynomials.

Remark 1.111. Consider lenses between polynomials of the form Sy^S. In terms of arenas, the set of options for each decision $s \in S$ is again just the set S of decisions. So decisions are all about "which $s \in S$ you're at, and which one you want to be at next".

A lens, i.e. a polynomial map $(\text{get}, \text{set}) \colon Sy^S \to Ty^T$ is as usual a way to delegate S-decisions to T-decisions. Among them, some lenses are considered to be "very well behaved" because they give T a more sensible form of control. Namely, they are the ones that satisfy the following for all $s \in S$ and $t, t' \in T$:

$$\text{get}(\text{set}(s,t)) = t \qquad \text{set}(s, \text{get}(s)) = s \qquad \text{set}(\text{set}(s,t), t') = \text{set}(s, t')$$

We will see these emerge from more general theory in **??**.

1.3 Dynamical systems using Poly

Let's start putting all this **Poly** stuff to use.

1.3.1 Moore machines

Remember Moore machines from **??**?

Definition 1.112. If A, B, and A are sets, an (A, B)-Moore machine with states S consists of two functions

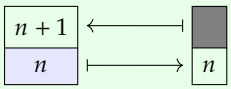
$$\text{expose}: S \to B$$
$$\text{update}: S \times A \to S$$

Look familiar? It's easy to see that an (A, B) Moore machine with states S is just a map of polynomials

$$Sy^S \to By^A$$

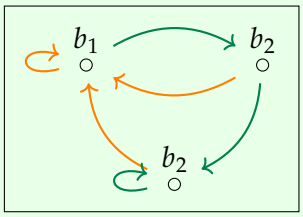
or to say it another way, a lens from $(S, S) \to (A, B)$ in the sense of Eq. (1.110).

Example 1.113. There is a dynamical system that takes unchanging input and produces as output the sequence of natural numbers $0, 1, 2, 3, \ldots$. It is a Moore machine of type $(1, \mathbb{N})$ with states $\mathbb{N}$. The polynomial map $\mathbb{N}y^{\mathbb{N}} \to \mathbb{N}y$ is given by identity on positions and $n \mapsto n + 1$ on directions



We can generalize both Sy^S and By^A in Definition 1.112, though the latter is much simpler and we'll spend much of the section on it. After some examples, we'll briefly preview something we're going to spend a lot of time later, namely what's special about Sy^S; then we'll discuss how to model regular languages using Moore machines and how products of polynomials allow multiple interfaces to act on the same system.

Example 1.114. Here's a picture of a Moore machine with $S := 3$ states:



Each state is labeled by its exposed value, i.e. element of $B := \{b_1, b_2\}$. Each state has two outgoing arrows, one orange and one green, so $A := \{\text{orange}, \text{green}\}$.

You can imagine barking "orange! orange! green! orange!" etc. at this machine, and it running through states.

Example 1.115. Here's a(n infinite) Moore machine with states $\mathbb{R}^2$:

$$\mathbb{R}^2 y^{\mathbb{R}^2} \to \mathbb{R}^2 y^{[0,1]\times[0,2\pi)}$$

Its output type is $\mathbb{R}^2$, which we might think of as a position, and its input type is $[0,1] \times [0,2\pi]$, which we might think of a command to move a certain amount in a certain direction. The map itself is given by the lens:

$$\begin{array}{cc} \mathbb{R}^2 \xrightarrow{\text{get}} \mathbb{R}^2 & \mathbb{R}^2 \times [0,1] \times [0,2\pi] \xrightarrow{\text{put}} \mathbb{R}^2 \\ (a,b) \mapsto (a,b) & (a,b,r,\theta) \mapsto (a + r\cos\theta, b + r\sin\theta) \end{array}$$

Example 1.116 (From functions to Moore machines). For any function $f : A \to B$ there is a corresponding (A,B)-Moore machine with states A that takes in a stream of A's and outputs the stream of B's obtained by applying f. It is given by the map $(f, f^\sharp) : Ay^A \to By^A$ that on positions is f, and for each $a \in A$ the on-directions map $f^\sharp_a : A \to A$ is the identity on A.

Exercise 1.117. For any function $f : A \to B$ there is a corresponding (A,B)-Moore machine with states B that takes in a stream of A's and outputs the stream of B's obtained by applying f. What is it? ◊

Exercise 1.118. Find $A, B \in$ **Set** such that the following can be identified with a morphism $Sy^S \to By^A$:

1. a *discrete dynamical system*, i.e. a set S and a function $S \to S$.
2. a *magma*, i.e. a set S and a function $S \times S \to S$.
3. a set S and a subset $S' \subseteq S$. ◊

Exercise 1.119. Consider the Moore machine in Example 1.115, and think of it as a robot. Using the terminology from that example, modify the robot as follows.

Add to its state a "health meter", which has a value between 0 and 10. Make the robot lose health whenever the x-coordinate of its position is negative. Use its health h as a multiplier, allowing it to move a distance of hr given an input of r. ◊

Exercise 1.120. Let's say a file of length n is a function $f : \mathsf{n} \to \mathsf{ascii}$, where $\mathsf{ascii} := 256$. Suppose given such a file and refer to elements of $\mathsf{n} = \{1, \dots, n\}$ as positions. Make a robot (Moore machine) whose output type is ascii and whose input type is

$$\{(s,t) \mid 1 \le s \le t \le n\} + \{\text{continue}\}$$

Given an input, if it is of the form (s, t) then the robot goes to position s in the file and begins reading (assuming it receives the "continue" signal) one character at a time, until it reaches position t. Then it continually outputs "done" until it receives another (s, t) pair. ◊

When we get to generalized Moore machines, we will be able to let the robot "close its port", so that while it can't receive signals while it's busy reading; see Example 1.138.

Exercise 1.121 (Tape of a Turing machine). A Turing machine has a tape. The tape has a position for each integer, and each position holds a value $v \in V = \{0, 1, -\}$ of 0,1, or blank. At any given time the tape not only holds this function $f : \mathbb{Z} \to V$ from positions to values, but also a distinguished choice $c \in \mathbb{Z}$ of "current" position. Thus the set of states of the tape is $T := V^{\mathbb{Z}} \times \mathbb{Z}$.

The Turing machine interacts with the tape by asking for the value at the current position, an element of V, and by telling it to change the value there as well as whether to move left or right. Thus the output of the tape is V and the input is $V \times \{L, R\}$.

1. Give the form of the tape as a Moore machine, i.e. map of polynomials $t : Sy^S \to p$ for appropriate $S \in S$ and $p \in \mathbf{Poly}$.
2. Write down the specific t that makes it act like a tape as specified above. ◊

The polynomial Sy^S as a comonad on Set. A *comonad* on **Set** is a functor $F : \mathbf{Set} \to \mathbf{Set}$, equipped with two natural transformations $\epsilon : F \to \mathrm{id}$ and $\delta : F \to F \circ F$, satisfying three equations. We don't need this now, so we won't get into it here. But we will note that every comonad comes from an adjunction, and the adjunction corresponding to Sy^S is

$$\mathbf{Set} \underset{\leftarrow_{-^S}}{\overset{-\times S}{\underset{\Rightarrow}{\longrightarrow}}} \mathbf{Set}$$

the "curry/uncurry" adjunction. In functional programming, the comonad Sy^S is called the *state comonad*,[3] and the elements of S are called states. It is no coincidence that we also refer to elements of S as states.

Again, we will be *very* interested in polynomial comonads later—as mentioned in Proposition 1.18 they are exactly categories!!—but for now we move on to things we can use right away in our story about dynamical systems.[4]

Regular languages Regular languages are very important in computer science. One way to express what they are is to say that they are exactly the languages recognizable by a deterministic finite state automaton. What is that?

[3] The comonad Sy^S is sometimes called the *store* comonad.

[4] If you're curious what category the comonad Sy^S corresponds to, it's the one with object set S and a unique morphism $s_1 \to s_2$ for every pair of objects $s_1, s_2 \in S$.

Definition 1.122. A *deterministic finite state automaton* consists of

1. a finite set S, elements of which are called *states*
2. a finite set A, elements of which are called *input symbols*,
3. a function $u : S \times A \to S$, called the *update function*,
4. an element $s_0 \in S$, called the *initial state*,
5. a subset $F \subseteq S$, called the *accept states*.

Proposition 1.123. A deterministic finite state automaton can be identified with a pair of maps

$$y \to Sy^S \to 2y^A.$$

Proof. A map $y \to Sy^S$ can be identified with an element $s_0 \in S$. A map $Sy^S \to 2y^A$ consists of a function $S \to 2$—which can be identified with a subset of S—together with a function $S \times A \to S$, which is the rest of the required structure. □

Products: two interfaces operating on the same system. One thing we can use right away in our thinking about dynamical systems is products of polynomials. The universal property of products says that for any polynomials s, p_1, p_2 and maps $s \to p_1$ and $s \to p_2$ there is a unique map $s \to p_1 p_2$. In the context of Moore machines, we have the following.

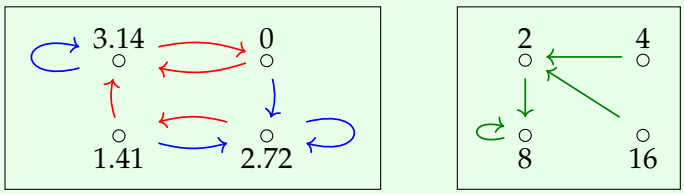
Proposition 1.124. Suppose given an (A_1, B_1)-Moore machine and an (A_2, B_2)-Moore machine, each with state set S. Then there is an induced (A_1+A_2, B_1B_2)-Moore machine, again with state set S.

Proof. We are given maps of polynomials $Sy^S \to B_1 y^{A_1}$ and $Sy^S \to B_2 y^{A_2}$. Hence by the universal property of products we have a map

$$Sy^S \to (B_1 y^{A_1}) \times (B_2 y^{A_2}) \cong (B_1 B_2) y^{A_1+A_2},$$

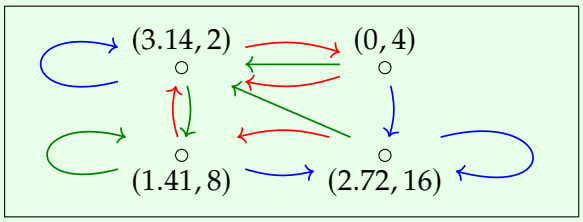
as desired. □

Example 1.125. Consider two four-state dynamical systems $4y^4 \to \mathbb{R}y^{\{r,b\}}$ and $4y^4 \to \mathbb{R}y^{\{g\}}$, 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.



The universal property of products provides a unique way to put these systems

together to obtain a morphism $4y^4 \to (\mathbb{R}y^{\{r,b\}} \times \mathbb{R}y^{\{g\}}) = (\mathbb{R}^2)y^{\{r,b,g\}}$. With the examples above, it looks like this:



In other words, if there are two or more interfaces that drive a single set of states, we can combine them.

Exercise 1.126 (Toward event-based systems). Let $f : Sy^S \to By^A$ be a Moore machine. It is constantly needing input at each time step. An event-based system is one that doesn't always get input, and only reacts when it does.

So suppose we want to allow our machine not to do anything. That is, rather than needing to press a button in A at each time step, we want to be able to *not* press any button, in which case the machine just stays where it is. We want a new machine $f' : Sy^S \to p$ that has this behavior; what is p and what is f'? ◊

1.3.2 Dependent Systems

Everything we've done above was for interfaces of the form By^A, i.e. for monomials. But the theory works just as well for an arbitrary p, a sum of monomials.

Definition 1.127 (Dependent systems)**.** A *dependent system* is a map of polynomials

$$Sy^S \to p$$

for some $S \in$ **Set** and $p \in$ **Poly**. The set S is called the set of *states* and the polynomial p is called the *(poly-) interface*.

We could also call these *generalized Moore machines*, since Moore machines were seen to be given by the special case $p = By^A$ for sets A, B. A polynomial might be $p = B_1y^{A_1} + \cdots + B_ky^{A_k}$ (though it doesn't have to be finite), so the output is an element of the coproduct $B_1 + \cdots + B_k$, and the choice of output determines what sort of input you get. What kind of output is that?

We now begin to think of outputs not only as outward expressions, but as positions that one takes within ones arena, terminology we've been using all along. If the arena is your body, this would be positions of your body. This includes where you go, as well as the direction you're looking with your head and eyes, whether your lips are pursed or not, etc. In fact, all talking and gesturing is performed by changing your position. And your position determines what inputs you'll notice: if your eyes are closed, your input type is different than if your eyes are open.

The position you're in is a sensory organ, a hand outstretched, an eye open or closed f.

If we squint, we could even see an output more as a sensing apparatus than anything else. This is pretty philosophical, but imagine your outputs—what you say and do—are more there as a question to the world, a way of sensing what the world is like.

But however you think of dependent systems, we need to get a feel for how they work. Let's start with something familiar.

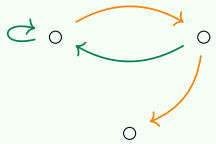
Example 1.128. Recall regular languages from Definition 1.122. Often one does not want to "keep going" after recognizing a word in ones language. For that, rather than use a map $Sy^S \to 2y^A$, we could use a map

$$(f_1, f^\sharp)\colon Sy^S \to y^A + 1$$

To give such a map, one in particular provides a function $f_1\colon S \to 2$; here we are thinking of 2 as the set of positions in $y^A + 1$. A function f_1 sends some elements of S to 1 and sends others to 2; those that are sent to 2 are said to be "accepted". This is captured by the fact that there are no options in the term 1, no inputs available there. In other words, the function $f^\sharp$ is trivial there.

On the other hand, $f_s^\sharp$ is not trivial on those elements $s \in S$ for which $f_1(s) = 1$. They must come equipped with a function $f_s^\sharp\colon A \to S$, saying how the machine updates on each element of A, starting at state s. Again, this is in line with the way state machines encode regular languages.

Exercise 1.129. Consider the deterministic finite state automaton shown below



The state without any outgoing arrows is the "accept" state for a regular language, and the left-most state is the start state. Answer the following questions, in keeping with the notation from Example 1.128.

1. What is S?
2. What is A?
3. In terms of regular languages, what is the alphabet here?
4. Specify the morphism of polynomials $Sy^S \to y^A + 1$.
5. Name a word that is accepted by this machine.
6. Name a word that is not accepted by this machine. ◊

Example 1.130 (Graphs). Given a graph $G = (E \rightrightarrows V)$ with source map $s : E \to V$ and target map $t : E \to V$, there is an associated polynomial

$$g := \sum_{v \in V} y^{s^{-1}(v)}.$$

We call this the *emanation polynomial of* G.

The graph itself can be seen as a dynamical system $f : Vy^V \to g$, where $f_1 = \mathrm{id}_V$ and $f^\sharp(v, e) = t(e)$.

Exercise 1.131. Pick your favorite graph G, and consider the associated dynamical system as in Example 1.130. Draw the associated labeled transition system as in Example 1.125. ◊

Example 1.132 (Adding a pause button). Given any dependent dynamical system $f : Sy^S \to p$, we can "add a pause button", meaning that for any state (and any position), we add an input that keeps the state where it is.

To do this, note that we have a map $\epsilon : Sy^S \to y$ given by identity (see **??**). By the universal property of products, we can pair f and ϵ to get a map $(f, \epsilon) : Sy^S \to py \cong p \times y$. This process is actually universal in a way we'll find important later. Indeed, it's called *copointing*; see Proposition 2.153.

Example 1.133. Suppose given a dependent dynamical system f that sometimes outputs an element of A and sometimes outputs only the hum. That is, its interface is $Ay + y$. What if we want to turn that into a dynamical system that always outputs an A by constantly repeating the last thing output by f. How do we construct it?

The output to our whole system is supposed to be Ay, and we already have $f : Sy^S \to Ay + y$. It will be enough to construct a map $g : (Ay + y) \otimes Ay^A \to Ay$, because then we can use

$$(SAy^{SA}) \cong (Sy^S) \otimes (Ay^A) \xrightarrow{f \otimes \mathrm{id}} (Ay + y) \xrightarrow{g} Ay$$

as our overall dynamical system. Since $\otimes$ distributes over $+$, we need to give $Ay \otimes Ay^A \to Ay$ and $y \otimes Ay^A \to Ay$. For the latter we use the identity, which we called $\epsilon : Ay^A \to y$ and for the former we use

$$Ay \otimes Ay^A \cong (Ay \otimes y^A) \otimes Ay \xrightarrow{\epsilon \otimes \mathrm{id}} y \otimes Ay$$

This accomplishes what we wanted.

Example 1.134 (Inputting a start state). Suppose you have a closed system $f^\sharp\colon Sy^S \to y$. The modeler can choose a start state $y \to Sy^S$, but what if we want some other system to choose the start state? We haven't gotten to wiring diagrams yet, but the idea is to create a system that starts as not-closed—accepting as input a state $s \in S$—and then dives into its closed loop with that start state.

Let $S' \coloneqq S + 1$, so that the start state $y \to S'y^{S'}$ now is canonical: it's the new 1. We also have a canonical inclusion $S \xrightarrow{i} S'$. We will give a morphism

$$S'y^{S'} \to y + y^S$$

that starts out with its outer box in the mode y^S of accepting an S-input, and then moves to the mode y so that it is a closed system forever after.

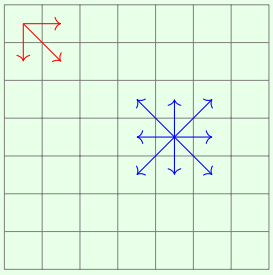
To give a morphism $S'y^{S'} \to y + y^S$, it is sufficient to give two morphisms: $Sy^{S'} \to y$ and $y^{S'} \to y^S$. The first is equivalent to a function $S \to S'$ and we take the map $S \xrightarrow{f^\sharp} S \xrightarrow{i} S'$; this means that whenever we want to update the state from a state in S we'll just do whatever our original closed system did. The second is also equivalent to a function $S \to S'$ and we use i; this means that whatever state is input at the beginning will be what we take as our first noncanonical state.

Exercise 1.135. Find what you think is an interesting generalization of deterministic finite state automata that you can model using generalized Moore machines. ◊

Example 1.136. Choose $n \in \mathbb{N}$, a *grid size*, and for each $i \in \mathsf{n} = \{1, \dots, n\}$ let $d(i)$ be the set

$$d(i) \coloneqq \begin{cases} \{0, 1\} & \text{if } i = 1 \\ \{-1, 0\} & \text{if } i = n \\ \{-1, 0, 1\} & \text{if } 1 < i < n \end{cases}$$

So $d(i)$ is the set of directions someone could move (or not move) if at position i.



Then a generalized Moore machine of the form $Sy^S \to p$, where

$$p = \sum_{(i,j)\in \mathsf{n}\times\mathsf{n}} y^{d(i)\times d(j)},$$

is one that has more movement options when it is in the center of the grid than when it is on the sides or corners.

Exercise 1.137. Add to Example 1.136 as follows.

1. Redefine p so that at each grid value, the robot can receive not only the set of directions it can move in but also a "reward value" $r \in \mathbb{R}$.
2. Define the set S of robot states so that an element $s \in S$ includes both the robot's position and a list of all reward values so far.
3. Define a morphism of polynomials $Sy^S \to p$ in a way that respects positions and properly updates the robots list of rewards, but otherwise does anything you want. ◊

Example 1.138. In Exercise 1.120 one is tasked with making a file reader $\mathsf{FileReader}$, where a file is a function $f : \mathsf{n} \to \mathsf{ascii}$. Now we take that same idea but make the robot have a different interface when it is in read-mode: namely, one where it cannot take in signals.

Let $\mathsf{State}_{\mathsf{FileReader}} := \{(s,t) \mid 1 \le s \le t \le n\}$ consist of a current position s and a terminal position t. For our interface, we'll have two modes, each of which exposes an ascii character:

$$\mathsf{Out}_{\mathsf{FileReader}} = \langle \mathrm{Accepting}(c), \mathrm{Busy}(c) \mid c \in \mathsf{ascii} \rangle$$

For our input, we need a family $\mathsf{In}_{\mathsf{FileReader}} : \mathsf{Out}_{\mathsf{FileReader}} \to \mathbf{Set}$. We'll define this by cases:

$$\mathsf{In}_{\mathsf{FileReader}}(\mathrm{Accepting}(c)) = \mathsf{State}_{\mathsf{FileReader}},$$
$$\mathsf{In}_{\mathsf{FileReader}}(\mathrm{Busy}(c)) = 1.$$

Our file reader will be Accepting if its current position is the terminal position; otherwise, it will be Busy. In either case, it will expose the ascii character at the current position.

$$\mathrm{expose}_{\mathsf{FileReader}}(s,t) = \begin{cases} \mathrm{Accepting}(f(s)) & \text{if } s = t \\ \mathrm{Busy}((f(s))) \end{cases}$$

While the file reader is Busy, it will step forward through the file. When it is

Accepting, it will set its new current and terminal position to be the input.

$$\text{update}_{\text{FileReader}}(s,t) = \begin{cases} \_ \mapsto (s+1,t) & \text{if expose}_{\text{FileReader}}(s,t) \text{ is Busy} \\ (s',t') \mapsto (s',t') & \text{if expose}_{\text{FileReader}}(s,t) \text{ is Accepting} \end{cases}$$

Let $A \coloneqq \{(s,t) \mid 1 \leq s \leq t \leq n\}$, and let $p \coloneqq \mathsf{ascii}\cdot y^A + \mathsf{ascii}\cdot y^1$; we construct a morphism in **Poly**

$$(r_1, r^\sharp) \colon Ay^A \to \mathsf{ascii}\cdot y^A + \mathsf{ascii}\cdot y^1$$

as follows.

$$A + B = \langle \text{inl}(a), \text{inr}(b) \mid a \in A, b \in B \rangle$$

$$A +_C B = \langle \text{inl}(a), \text{inr}(b) \mid a \in A, b \in B, \forall c \in C. \text{inl}(f(c)) = \text{inr}(g(c)) \rangle$$

On positions define

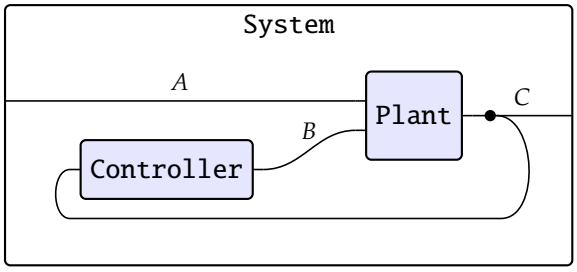
$$r_1(s,t) \coloneqq \begin{cases} \texttt{inl}\ f(s) & \text{if } s = t \\ \texttt{inr}\ f(s) & \text{if } s < t \end{cases}$$

$$r^\sharp_{(t,t)}(s',t') \coloneqq (s',t') \qquad r^\sharp_{(s,t)}(1) \coloneqq (s+1,t)$$

Exercise 1.139. Make a file reader that acts like that in Example 1.138, except that it only emits output $o \in \mathsf{ascii}$ when $o = 100$. ◊

1.3.3 Wiring diagrams

We want our dynamical systems to interact with each other.



(1.140)

In this picture the plant is receiving information from the world outside the system, as well as from the controller. It's also producing information for the outside world which is being monitored by the controller.

There are three boxes shown in (1.140): the controller, the plant, and the system. Each has inputs and outputs, and so we can consider the interface as a monomial.

$$\texttt{Plant} = Cy^{AB} \qquad \texttt{Controller} = By^C \qquad \texttt{System} = Cy^A. \tag{1.141}$$

The wiring diagram itself is a morphism in **Poly** of the form

$$w\colon \texttt{Plant} \otimes \texttt{Controller} \to \texttt{System}$$

Since everything involved is a monomial—the parallel product of monomials is a monomial—the whole wiring diagram w is a lens $CBy^{ABC} \to Cy^{A}$. This morphism says how wires are feeding from outputs to inputs. Like all lenses, it consists of two functions

$$\text{get}\colon CB \to C \qquad \text{and} \qquad \text{put}\colon CBA \to ABC$$

The first says "inside the system you have boxes outputting values of type C and B. The system needs to produce an output of type B; how shall I obtain it?" The second says "the system is providing an input value of type A, and inside the system you have boxes outputting values of type C and B. These boxes need input values of type A, B, and C; how shall I obtain them?" The answer of course is that *get* is given by projection $(c, b) \mapsto c$ and *put* is given by a permutation $(c, b, a) \mapsto (a, b, c)$. The wiring diagram is a picture that tells us which maps to use.

Exercise 1.142.

1. Make a new wiring diagram like (1.140) except where the controller also receives information of type A' from the outside world.
2. What are the monomials in your diagram (replacing (1.141))?
3. What is the morphism of polynomials corresponding to this diagram? ◊

Now suppose given a dynamical system in each inner box:

$$Sy^{S} \xrightarrow{f} \texttt{Plant} \qquad \text{and} \qquad Ty^{T} \xrightarrow{g} \texttt{Controller}$$

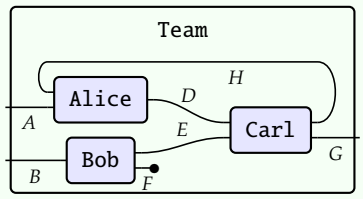
Then since $\otimes$ is a monoidal product on **Poly** (see Proposition 1.97), we get a map

$$Sy^{S} \otimes Ty^{T} \xrightarrow{f \otimes g} \texttt{Plant} \otimes \texttt{Controller}$$

In other words we have a morphism of polynomials $STy^{ST} \to \texttt{Plant} \otimes \texttt{Controller}$; that's a new dynamical system with state space $ST = (S \times T)$; a state in it is just a pair of states, one in S and one in T. Furthermore our wiring diagram already gave us a map $\texttt{Plant} \otimes \texttt{Controller} \to \texttt{System}$, so combining, we have a new system

$$STy^{ST} \to \texttt{System}.$$

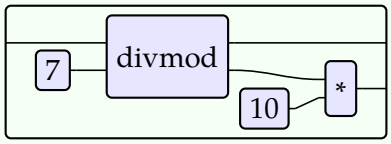
Exercise 1.143. Consider the following wiring diagram.



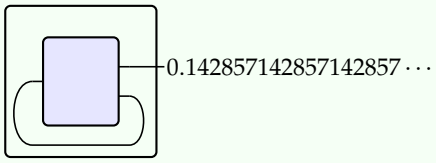
1. Write out the polynomials for each of Alice, Bob, and Carl.
2. Write out the polynomial for the outer box, Team.
3. The wiring diagram constitutes a morphism f in **Poly**; what is its type $f : ? \to ?$
4. What morphism is it?
5. Suppose we are given dynamical systems $Ay^A \to \texttt{Alice}$, $By^B \to \texttt{Bob}$, and $Cy^C \to \texttt{Carl}$. What is the induced dynamical system on `Team`? ◊

Exercise 1.144 (Long division).

1. Come up with a function "divmod" of type $\mathbb{N} \times \mathbb{N}_{\geq 1} \to \mathbb{N} \times \mathbb{N}$ and which, for example, sends $(10, 7)$ to $(1, 3)$ and $(30, 7)$ to $(4, 2)$.
2. Use Exercise 1.117 to turn it into a dynamical system.
3. Interpret the following wiring diagram:



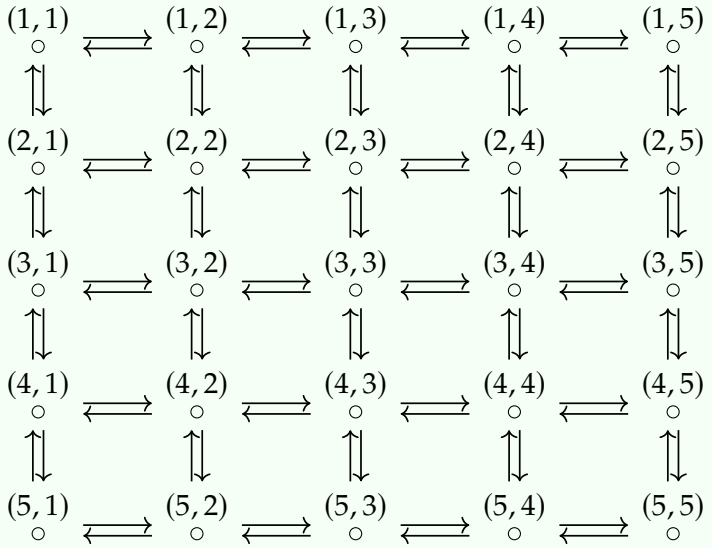
4. Use the above and a diagram of the following form to create a function that spits out the base-10 digits of 1/7.



◊

Exercise 1.145 (Cellular automata). Let $G = (V, E)$ be a simple graph, i.e. $V, E \in$ **Set**

and $E \subseteq V \times V$. You can imagine it as a grid



finite or infinite, or just an arbitrary graph. For every vertex v, the set of vertices v' for which $(v', v) \in E$, i.e. for which there is an edge $v' \to v$, is denoted

$$I(v) \coloneqq \{v' \mid (v', v) \in E\}.$$

For each $v \in V$, let $p_v \coloneqq \mathsf{2}y^{2^{I(v)}}$; it "outputs" a color $\mathsf{2} \cong \{\texttt{black}, \texttt{white}\}$ and inputs a function $I(v) \to \mathsf{2}$, specifying what all the neighbors are outputting.

1. In the drawn grid, what is $I(1,1)$? What is $I(2,2)$?
2. Specify a morphism $g\colon \bigotimes_{v \in V} p_v \to y$ that passes to each vertex v the colors of its neighbors in $I(v)$.
3. Suppose that for each vertex $v \in V$ you are given a function $f_v\colon \mathsf{2}^{I(v)} \to \mathsf{2}$. Use it to construct a dynamical system $f'_v\colon \mathsf{2}y^{\mathsf{2}} \to p_v$ that updates its state in keeping with f_v and outputs its state directly.
4. Briefly look up cellular automata in a reference of your choice. Would you say that the dynamical system $\bigotimes_{v \in V} \mathsf{2}y^{\mathsf{2}} \xrightarrow{\bigotimes f'_v} \bigotimes_{v \in V} p_v \xrightarrow{g} y$ we obtain by wiring together the dynamical systems in the specified way does the same thing as the cellular automata in your reference? ◊

1.3.4 General interaction

In general, we want systems that can change their interface—remove a port, add a port, change the type of a port, etc.—based on their internal states. But when such systems interact with others, the interaction pattern must be able to accommodate all of the various combinations of interfaces.

Example 1.146. Suppose given two interfaces p and p', having mode sets M and M'

respectively

$$p \coloneqq \sum_{m \in M} B_m y^{A_m} \qquad \text{and} \qquad p' \coloneqq \sum_{m' \in M'} B'_m y^{A'_m}$$

The parallel product of these is:

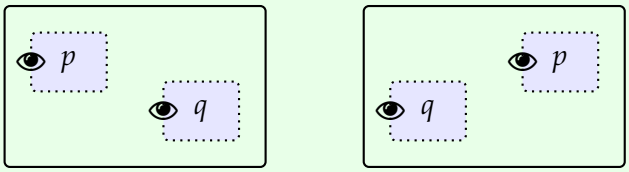
$$p \otimes p' \cong \sum_{(m,m') \in MM'} B_m B'_m y^{A_m A'_m}$$

so the interface $B_m B'_{m'} y^{A_m A'_{m'}}$ at each of these $(M \times M')$-many modes (m, m') must be accommodated in any morphism $w \colon p \otimes p' \to q$. For example if $M = 2$ and $M' = 3$ then w can be specified by six maps.

But in fact the possibilities for interaction are much more general than we have led the reader to believe. They may not be broken down into modes at all.

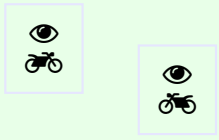
Example 1.147. Let $p \coloneqq By^A$ and $p' = B'y^{A'}$. To give a morphism $f \colon p \otimes p' \to y$, one specifies a map $B \times B' \to 1$, which is no data, as well as a map $BB' \to AA'$. In other words, for every pair of outputs (b, b') one specifies a pair of inputs (a, a').

Let's think of elements of B and B' not as outputs, but as positions.



Then given both positions (b, b'), the interaction pattern f tells us what the two eyes see, i.e. what values of (a, a') we get.

Example 1.148. Suppose you have two systems p, q each of type $p, q \coloneqq \mathbb{R}^2 y^{\mathbb{R}^2 - (0,0)}$.



Taking all pairs of reals except $(0, 0)$ corresponds to the fact that the eye cannot see that which is at the same position as the eye.

Let's have the two systems constantly approaching each other with a force equal to the reciprocal of the squared distance between them. If they finally collide, let's have the whole thing come to a halt.

To do this, we want the outer system to be of type $\{\texttt{go}\}y + \{\texttt{stop}\}$. The morphism

$\mathbb{R}^2 y^{\mathbb{R}^2-(0,0)} \otimes \mathbb{R}^2 y^{\mathbb{R}^2-(0,0)} \to \{\texttt{go}\}y + \{\texttt{stop}\}$ is given on positions by

$$\big((x_1, y_1), (x_2, y_2)\big) \mapsto \begin{cases} (\texttt{stop}, 1) & \text{if } x_1 = x_2 \text{ and } y_1 = y_2 \\ (\texttt{go}, 1) & \text{otherwise.} \end{cases}$$

On directions, we use the function

$$\big((x_1, y_1), (x_2, y_2)\big) \mapsto \big((x_2 - x_1, y_2 - y_1), (x_1 - x_2, y_1 - y_2)\big).$$

Now each system is able to see the vector pointing from it to the other system (unless that vector is zero, in which case the whole thing has halted). Let's use these vectors to define the internal dynamics of each system. Each system will hold as its internal state its current position and velocity, i.e. $S = \mathbb{R}^2 \times \mathbb{R}^2$. To define a map of polynomials $Sy^S \to \mathbb{R}^2 y^{\mathbb{R}^2-(0,0)}$ we simply output the current position and update the current velocity by adding a vector pointing to the other system and having appropriate magnitude:

$$\mathbb{R}^2 \times \mathbb{R}^2 \xrightarrow{\text{get}} \mathbb{R}^2$$
$$\big((x, y), (x', y')\big) \xmapsto{\text{get}} (x, y)$$

$$\mathbb{R}^2 \times \mathbb{R}^2 \times (\mathbb{R}^2 - (0,0)) \xrightarrow{\text{put}} \mathbb{R}^2 \times \mathbb{R}^2$$
$$\big((x, y), (x', y'), (a, b)\big) \xmapsto{\text{put}} \left(x + x', y + y', x' + \frac{a}{(a^2 + b^2)^{3/2}}, y' + \frac{b}{(a^2 + b^2)^{3/2}}\right)$$

Exercise 1.149. Let $p, q \coloneqq \mathbb{N}y^{\mathbb{N}}$.

1. Write a polynomial morphism $p \otimes q \to y$ that corresponds to the function $(a, b) \mapsto (b, a + b)$.
2. Write dynamical systems $\mathbb{N}y^{\mathbb{N}} \to p$ and $\mathbb{N}y^{\mathbb{N}} \to q$, each of which simply outputs the previous input.
3. Suppose each system starts in state $1 \in \mathbb{N}$. What is the trajectory of the p-system? ◊

Exercise 1.150. Suppose (X, d) is a metric space, i.e. X is a set and $d\colon X \times X \to \mathbb{R}$ is a function satisfying the usual laws. Let's have robots interact in this space.

Let A, A' be sets, each thought of as a set of signals, and let $a_0 \in A$ and $a_0' \in A'$ be elements, each thought of as a default value. Let $p \coloneqq AXy^{A'X}$ and $p' \coloneqq A'Xy^{AX}$, and imagine there are two robots, one with interface p and one with interface p'.

1. Write down a morphism $p \otimes p' \to y$ such that each robot receives the other's location, but that it only receives the other's signal when the locations x, x' are sufficiently close, $d(x, x') < 1$. Otherwise it receives the default signal.

2. Write down a morphism $p \otimes p' \to y^{[0,5]}$ where the value $s \in [0,5]$ is a scalar, allowing the signal to travel s times further.
3. Suppose that each robot has a set S, S' of private states. What functions are involved in providing a dynamical system $f\colon SXy^{SX} \to AXy^{A'X}$?
4. Change the setup in any way so that the robots only extend a port to hear the other's signal when the distance between them is less than 1. Otherwise, they can only detect the position (element of X) that the other currently inhabits. ◊

So what is a map $p_1 \otimes \cdots \otimes p_k \to q$ in general? It's a protocol by which he k-many participants p_i together decide what decision q must make, as well as how q's choice among its options (the decision once made) is passed back and distributed as an option at each p_i.

Example 1.151 (Cellular automata who vote on their interaction pattern). Recall from Exercise 1.145 how we constructed cellular automata on a graph $G = (V, E)$. Here $E \subseteq V \times V$, or equivalently what we might call an *interaction pattern* $I\colon V \to \mathsf{2}^V$, specifies the incoming neighbors $I(v)$ of each $v \in V$.

Suppose now that we are given a function $i\colon V \to \mathbb{N}$ that we think of as specifying the number $i(v)$ of neighbors each $v \in V$ accepts. Let $\mathsf{i}(v) = \{1, 2, \ldots, i(v)\}$. We will be interested in the polynomial $p_v := \mathsf{2}y^{\mathsf{2}^{\mathsf{i}(v)}}$ for each v; it represents an interface that outputs a color $\mathsf{2} \cong \{\texttt{black}, \texttt{white}\}$ and that inputs a function $\mathsf{i}(v) \to \mathsf{2}$, meant to give the colors of the neighboring vertices.

Say that an interaction pattern $I\colon V \to \mathsf{2}^V$ *respects* i if we have an isomorphism $I(v) \cong \mathsf{i}(v)$ for each $v \in V$. Suppose given a function $I\colon \mathsf{2}^V \to (\mathsf{2}^V)^V$ such that for each element $s \in \mathsf{2}^V$, the interaction pattern $I_s\colon V \to \mathsf{2}^V$ respects i. In the case of Exercise 1.145, I was a constant function. Now we can think of it like all the vertices are voting, via I, on the connection pattern.

We can put this all together by giving a morphism in **Poly** of the form

$$\bigotimes_{v \in V} p_v \cong \mathsf{2}^V y^{\mathsf{2}^{\sum_{v \in V} \mathsf{i}(v)}} \longrightarrow y. \tag{1.152}$$

We can such a morphism with a function $g\colon \mathsf{2}^V \to \mathsf{2}^{\sum_{v \in V} \mathsf{i}(v)}$. Suppose given $s \in \mathsf{2}^V$, so that we have an isomorphism $I_s(v) \cong \mathsf{i}(v)$ for each $v \in V$; we want a function $g(s)\colon \sum_{v \in V} I_s(v) \to \mathsf{2}$. That is, for each v we want a function

$$g(s)_v\colon I_s(v) \to \mathsf{2}.$$

But $I_s(v) \subseteq V$, so our function $s\colon V \to \mathsf{2}$ induces the desired function $I_s(v) \to \mathsf{2}$.

We have accomplished our goal: the automata vote on their connection pattern. Of course, we don't mean to imply that this vote needs to be democratic or fair in any way: it is an arbitrary function $I\colon \mathsf{2}^V \to (\mathsf{2}^V)^V$. It could be dictated by a given vertex $v_0 \in V$ in the sense that its on/off state completely determines the connection pattern

$V \times V \to 2$; this would be expressed by saying that I factors as $2^V \to 2^{v_0} \cong 2 \xrightarrow{I_0} (2^V)^V$ for some I_0.

Exercise 1.153. Change Example 1.151 slightly by changing the outer box.

1. First change it to Ay for some set A of your choice, and update (1.152) so that the system outputs some aspect of the current state 2^V.
2. What would it mean to change (1.152) to a map $\bigotimes_{v \in V} p_v \to y^A$ for some A? ◊

Example 1.154. Recall the picture from Example 1.8. We said that when too much force is applied to a material, bonds can break. Let's simplify the picture a bit.

We will imagine systems Φ_1 and Φ_2 as initially connected in space, that they experience forces from the outside world, and that—for as long as they are connected—they experience forces from each other. More precisely, each internal arena is defined by

$$p_1 = p_2 \coloneqq Fy^{FF} + y^F.$$

Elements of F will be called *forces*. We need to be able to add and compare forces, i.e. we need F to be an ordered monoid; let's say $F = \mathbb{N}$ for simplicity. The idea is that the arena has two modes: the monomial Fy^{FF} consisting of two input forces (one from its left and one from its right) and an output force f_i, and the monomial y^F consisting of one input force (just from the outside). Similarly, in the first mode the system Φ_i is outputting a force for the other—whether the other uses it or not—but in the second mode the system produces no force for the other.

The external arena is defined to be

$$p \coloneqq y^{FF};$$

it takes as input two forces (f_L, f_R) and produces unchanging output.

Though the systems Φ_1 and Φ_2 may be initially connected, if the forces on either one surpass a threshold, that system stops sending and receiving forces from the other. The connection is broken and neither system ever receives forces from the other again. This is what we will implement explicitly below.

To do so, we need to create a contract $p_1 \otimes p_2 \to p$ of the external arena p around (the arenas of) the internal systems. That is, we need to give a morphism of polynomials

$$\kappa \colon (Fy^{FF} + y^F) \otimes (Fy^{FF} + y^F) \to y^{FF}.$$

By distributivity and the universal property of coproducts, it suffices to give four maps:

$$\begin{aligned}\kappa_{11}\colon\ & FFy^{(FF)(FF)} \to y^{FF}\\ \kappa_{12}\colon\ & Fy^{(FF)F} \to y^{FF}\\ \kappa_{21}\colon\ & Fy^{F(FF)} \to y^{FF}\\ \kappa_{22}\colon\ & y^{FF} \to y^{FF}\end{aligned}$$

The middle two maps κ_{12} and κ_{21} won't actually occur in our dynamics, so we take them to be arbitrary. We take the last map κ_{22} to be an identity (the forces from outside are passed to the two internal boxes). The first map κ_{11} is equivalent to a function $(FF)(FF) \to (FF)(FF)$ which we take to be $((f_1, f_2), (f_L, f_R)) \mapsto ((f_L, f_2), (f_1, f_R))$.

Now that we have the arenas wired together, it remains to give the dynamics on the internal boxes. The states in the two cases will be identical, namely $S := F + 1$, meaning that at any point the system will either be in the state of holding a force or not. The dynamics will be identical as well, up to a symmetry swapping left and right; let's work with the first. Its interface is $p_1 = Fy^{FF} + y^F$ and its dynamics are given by

$$\Phi_1\colon (F+1)y^{F+1} \to Fy^{FF} + y^F$$

which splits up as the coproduct of $Fy^{F+1} \to Fy^{FF}$ and $y^{F+1} \to y^F$. The second map corresponds to when the connection is broken; it is given by projection, meaning it just updates the state to be the received force. The first map $Fy^{F+1} \to Fy^{FF}$ corresponds to the case where the system is holding some force, is receiving two input forces and must update its state and produce one output force. For the passforward $F \to F$, let's use identity meaning it outputs the force it's holding. For the passback $F(FF) \to \{\texttt{Just}\}F + \{\texttt{Nothing}\}$, let's use the map $(f, (f_L, f_2)) \mapsto t(f_L, f_2)$ defined here:

$$t(f_L, f_2) := \begin{cases} \texttt{Just}\ f_L & \text{if } f_1 + f_2 < 100\\ \texttt{Nothing} & \text{otherwise}\end{cases}$$

Thus when the sum of forces is high enough, the internal state is updated to the broken state; otherwise it is sent to the force it receives from outside.

Example 1.155. We want to consider the case of a company C that may change its supplier based on its internal state. The company has no output wires, but has two modes of operation—two positions—corresponding to who it wants to receive widgets W from:

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 $2y^W \otimes Wy \otimes Wy \to y$. Its on-positions function $2W^2 \to 1$ is uniquely determined, and its on-directions function $2W^2 \to W$ is the evaluation. In other words, the company's position determines which supplier from which it receives widgets.

Example 1.156. When someone assembles a machine, their own outputs dictate the connection pattern of the machine's components.

(1.157)

In order for the above picture to make sense, A has the same output that B has as input, say X, and we need a default value $x_0 \in X$ for B to input when not connected to A.

We could say that the person in (1.157) 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 $2y \otimes Xy \otimes y^X \to y$. The morphism $2Xy^X \to y$ is uniquely determined on positions, and on directions it is given by cases $(1, x) \mapsto x_0$ and $(2, x) \mapsto x$.

We can easily generalize Example 1.156. Indeed, we will see in the next section that there is a polynomial $[()q_1 \otimes \cdots \otimes q_k\ ,\ r)$ of all ways $q_1, \ldots, q_k$ can interact in r, and that a map from some p to it is just a bigger interaction pattern:

$$\mathbf{Poly}(p, [q_1 \otimes \cdots \otimes q_k\ ,\ r]) \cong \mathbf{Poly}(p \otimes q_1 \otimes \cdots \otimes q_k\ ,\ r).$$

In other words, if p thinks its deciding how $q_1, \ldots, q_k$ are wired up in r, and gets feedback from that wiring pattern itself, then in actuality p is just part of a wiring diagram with $q_1, \ldots, q_k$ inside of r.

What it also means is that if you want, you can put a little dynamical system inside of $[q_1 \otimes \cdots \otimes q_k, r]$ and have it constantly choosing interaction patterns. Let's see how it works.

1.3.5 Closure of $\otimes$

The parallel monoidal product is closed—we have a monoidal closed structure on **Poly**—meaning that there is an operation, which we denote $[-,-]\colon \mathbf{Poly}^{\mathrm{op}} \times \mathbf{Poly} \to \mathbf{Poly}$ and an isomorphism

$$\mathbf{Poly}(p \otimes q, r) \cong \mathbf{Poly}(p, [q, r])$$

natural in p, q, r. The closure operation is defined on q, r as follows:

$$[q, r] \coloneqq \prod_{j \in q(1)} r \circ (q[j]y) \tag{1.158}$$

Exercise 1.159. Show that there is an isomorphism

$$[q, r] \cong \sum_{f\colon p \to q} y^{\sum_{i \in q(1)} r[f(i)]}$$

where the sum is indexed by $f \in \mathbf{Poly}(p, q)$. ◊

Example 1.160. For any $A \in \mathbf{Set}$ we have

$$[y^A, y] \cong Ay \qquad \text{and} \qquad [Ay, y] \cong y^A.$$

These both follow directly from (1.158).

Exercise 1.161. Calculate $[q, r]$ for $q, r \in \mathbf{Poly}$ given as follows.

1. $q \coloneqq 0$ and r arbitrary.
2. $q \coloneqq 1$ and r arbitrary.
3. $q \coloneqq y$ and r arbitrary.
4. $q \coloneqq A$ and $r \coloneqq B$ for $A, B \in \mathbf{Set}$ (constants).
5. $q \coloneqq Ay$ and $r \coloneqq By$ for $A, B \in \mathbf{Set}$ (linears).
6. $q \coloneqq 2y^3 + y^2$ and $r \coloneqq 4y^2 + 7$. ◊

Exercise 1.162. Show that for any $p \in \mathbf{Poly}$, if there is an isomorphism $[[p, y], y] \cong p$, then p is either linear Ay or representable y^A for some A. Hint: first show that p must be a monomial. ◊

Proposition 1.163. With $[-,-]$ as defined in (1.158), there is a natural isomorphism

$$\mathbf{Poly}(p \otimes q, r) \cong \mathbf{Poly}(p, [q, r]). \tag{1.164}$$

Proof. We have the following chain of natural isomorphisms:

$$\begin{aligned} \mathbf{Poly}(p \otimes q, r) &\cong \mathbf{Poly}\Big(\sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i]q[j]}, r\Big) \\ &\cong \prod_{i \in p(1)} \prod_{j \in q(1)} \mathbf{Poly}(y^{p[i]q[j]}, r) \end{aligned}$$

$$\begin{aligned}
&\cong \prod_{i\in p(1)} \prod_{j\in q(1)} r(p[i]q[j]) \\
&\cong \prod_{i\in p(1)} \prod_{j\in q(1)} \mathbf{Poly}(y^{p[i]}, r\circ(q[j]y)) \\
&\cong \prod_{i\in p(1)} \mathbf{Poly}\Big(y^{p[i]}, \prod_{j\in q(1)} r\circ(q[j]y)\Big) \\
&\cong \mathbf{Poly}(p,[q,r]). \qquad \square
\end{aligned}$$

Exercise 1.165. Show that for any p, q we have an isomorphism of sets

$$\mathbf{Poly}(p,q) \cong [p,q](1).$$

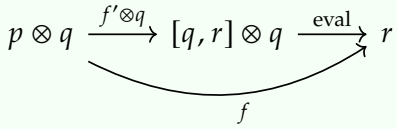
Hint: you can either use the formula (1.158), or just use (1.164) with the Yoneda lemma and the fact that $y \otimes p \cong p$. ◊

For any $p, q \in \mathbf{Poly}$ there is a canonical *evaluation* map

$$\text{eval}\colon [p,q]\otimes p \longrightarrow q. \tag{1.166}$$

Exercise 1.167.

1. Obtain the evaluation map $\text{eval}\colon [p,q]\otimes p \longrightarrow q$ from (1.164).
2. Show that for any p, q, r and map $f\colon p\otimes q \to r$, there is a unique morphism $f'\colon p \to [q,r]$ such that the following diagram commutes



◊

Exercise 1.168.

1. For any S, obtain the map $Sy^S \to y$ whose on-directions map is identity, using eval and Example 1.160.
2. Show that maps of the four types κ_{11}, κ_{12}, κ_{21}, κ_{22} shown in Example 1.154 can be obtained by tensoring together identity maps and eval maps. ◊

Example 1.169 (Modeling your environment without knowing what it is). Let's imagine a robot whose interface is an arbitrary polynomial p. Let's imagine it is living together in a closed system

$$f\colon (q_1\otimes\cdots\otimes q_n)\otimes p \to y$$

with some other robots whose interfaces are $q_1, \ldots, q_n$; let $q \coloneqq (q_1 \otimes \cdots \otimes q_n)$. The interaction pattern induces a morphism $f' \colon q \to [p, y]$ such that the original system f factors through the evaluation $[p, y] \otimes p \to y$.

In other words $[p, y]$ holds within it all of the possible ways p can interact with other systems in a closed box.[a] To investigate this just a bit, note that $[p, y] \cong \prod_{i \in p(1)} p[i]y$. That is, for each position in p it produces a direction there, which is just what p needs as input.

Now suppose we were to populate the interface p with dynamics, a map $Sy^S \to p$. One could aim to choose a set S along with an interesting map $g \colon S \to \mathbf{Poly}(p, y)$. Then each state s would include a guess $g(s)$ about what the state of the environment is in. This is not the real environment q, but just the environment as it affects p, namely $[p, y]$. The robot's states model environmental conditions.

[a] And if you want the generic way p to interact with other systems in a box r, just use $[p, r]$.

Exercise 1.170. Show that for any polynomials p_1, p_2, q, we have an isomorphism

$$[p_1 + p_2, q] \cong [p_1, q] \times [p_2, q].$$

◊

Example 1.171 (Chu &). Suppose given polynomials $p_1, p_2, q_1, q_2, r \in \mathbf{Poly}$ and morphisms

$$\varphi_1 \colon p_1 \otimes q_1 \to r \qquad \text{and} \qquad \varphi_2 \colon p_2 \otimes q_2 \to r$$

One might call these "r-Chu spaces". One operation you can do with these as Chu spaces is to return something denoted $\varphi_1 \& \varphi_2$, or "φ_1 *with* φ_2" of the following type:

$$\varphi_1 \& \varphi_2 \colon (p_1 \times p_2) \otimes (q_1 + q_2) \to r$$

Suppose given a position in p_1 and a position in p_2. Then given a position in either q_1 or q_2, one evaluates either φ_1 or φ_2 respectively to get a position in r; given a direction there, one returns the corresponding direction in q_1 or q_2 respectively, as well as a direction in $p_1 \times p_2$ which is either a direction in p_1 or in p_2.

This sounds complicated, but it can be done formally, once we have monoidal closure. We first rearrange both φ_1, φ_2 to be p-centric, using monoidal currying:

$$\psi_1 \colon p_1 \to [q_1, r] \qquad \text{and} \qquad \psi_2 \colon p_2 \to [q_2, r]$$

Now we multiply to get $\psi_1 \times \psi_2 \colon p_1 \times p_2 \to [q_1, r] \times [q_2, r]$. Now we use Exercise 1.170 to see that $[q_1, r] \times [q_2, r] \cong [q_1 + q_2, r]$, and then finally monoidal-uncurry to obtain $p_1 \times p_2) \otimes (q_1 + q_2) \to r$ as desired.

1.4 Bonus math about the category Poly

The category **Poly** has very useful formal properties, including completion under colimits and limits, various adjunctions with **Set**, factorization systems, and so on. Most of the following material is not necessary for the development of our main story, but we collect it here for reference. The reader can skip directly to Chapter 2 if so inclined. Better yet might be to just gently leaf through Section 1.4, to see how well-behaved and versatile the category **Poly** really is.

1.4.1 Special polynomials and adjunctions

There are a few special classes of polynomials that are worth discussing:

1. constant polynomials $0, 1, 2, A$;
2. linear polynomials $0, y, 2y, Ay$;
3. pure-powers polynomials, $1, y, y^2, y^A$; and
4. monomials $0, A, y, 2y^3, Ay^B$.

The first two classes, constant and linear polynomials, are interesting because they both put a copy of **Set** inside **Poly**, as we'll see in Propositions 1.173 and 1.174. The third puts a copy of $\mathbf{Set}^{\mathrm{op}}$ inside **Poly**, and the fourth puts a copy of bimorphic lenses inside **Poly**, as we saw in Section 1.2.4 (page 32).

Exercise 1.172. Which of the four classes above are closed under

1. the cocartesian monoidal structure $(0, +)$?
2. the cartesian monoidal structure $(1, \times)$?
3. the parallel monoidal structure $(y, \otimes)$?
4. composition of polynomials $p \circ q$?[a] ◊

[a] Composition, together with the unit y, is in fact yet another monoidal structure, as we will see in Chapter 2.

Proposition 1.173. There is a fully faithful functor **Set** $\to$ **Poly** sending $A \mapsto Ay^0 = A$.

Proof. By Eq. (1.59), a map $f : Ay^0 \to By^0$ consists of a function $f : A \to B$ and for each $a \in A$ a function $0 \to 0$. There is only one such function, so f can be identified with just a map of sets $A \to B$. □

Proposition 1.174. There is a fully faithful functor **Set** $\to$ **Poly** sending $A \mapsto Ay$.

Proof. By Eq. (1.59), a map $f : Ay^1 \to By^1$ consists of a function $f : A \to B$ and for each $a \in A$ a function $1 \to 1$. There is only one such function, so f can be identified with just a map of sets $A \to B$. □

Theorem 1.175. **Poly** has an adjoint quadruple with **Set**:

$$\mathbf{Set} \;\begin{array}{c} \xleftarrow{p(0)} \\ \xrightarrow[A]{} \\ \xleftarrow[p(1)]{} \\ \xrightarrow[Ay]{} \end{array}\; \mathbf{Poly} \tag{1.176}$$

where the functors have been labeled by where they send $A \in \mathbf{Set}$ and $p \in \mathbf{Poly}$.

Both rightward functors are fully faithful.

Proof. For any set A, there is a functor $\mathbf{Poly} \to \mathbf{Set}$ given by sending p to $p(A)$; it is $\mathbf{Poly}(A, -)$. This, together with Propositions 1.173 and 1.174 give us the four functors and the fact that the two rightward functors are fully faithful. It remains to provide the following three natural isomorphisms:

$$\mathbf{Set}(A, p(0)) \cong \mathbf{Poly}(A, p) \qquad \mathbf{Set}(p(1), A) \cong \mathbf{Poly}(p, A) \qquad \mathbf{Set}(A, p(1)) \cong \mathbf{Poly}(Ay, p).$$

All three come from our main formula, Eq. (1.63); we leave the details to the reader in Exercise 1.177. □

Exercise 1.177. Here we prove the remainder of Theorem 1.175 using Eq. (1.63):

1. Provide a natural isomorphism $\mathbf{Set}(A, p(0)) \cong \mathbf{Poly}(A, p)$.
2. Provide a natural isomorphism $\mathbf{Set}(p(1), A) \cong \mathbf{Poly}(p, A)$.
3. Provide a natural isomorphism $\mathbf{Set}(A, p(1)) \cong \mathbf{Poly}(Ay, p)$. ◊

Exercise 1.178. Show that for any polynomial p, its set $p(1)$ of positions is in bijection with the set of functions $y \to p$. ◊

In Theorem 1.175 we see that $p \mapsto p(0)$ and $p \mapsto p(1)$ have left adjoints. This is true more generally for any set A in place of 0 and 1, as we show in Corollary 1.181. However, the fact that $p \mapsto p(1)$ is also a right adjoint—and hence that we have the *quadruple* of adjunctions in (1.176)—is special to $A = 0, 1$.

Next we note that the set of polynomial morphisms $p \to q$

Proposition 1.179. There is a two-variable adjunction between **Set**, **Poly**, and **Poly**:[a]

$$\mathbf{Poly}(Ap, q) \cong \mathbf{Poly}(p, q^A) \cong \mathbf{Set}(A, \mathbf{Poly}(p, q)). \tag{1.180}$$

[a] The first set $\mathbf{Poly}(Ap, q)$ involves the A-fold coproduct of p and the middle set $\mathbf{Poly}(p, q^A)$ involves the A-fold product of q, neither of which we have proven exists. For organizational purposes, we put that off until Proposition 1.192.

Proof. Since Ap is the A-fold coproduct of p and q^A is the A-fold product of q, the universal properties of coproduct and product give isomorphisms

$$\mathbf{Poly}(Ap, q) \cong \prod_{a \in A} \mathbf{Poly}(p, q) \cong \mathbf{Poly}(p, q^A).$$

The middle set is in bijection with $\mathbf{Set}(A, \mathbf{Poly}(p, q))$, completing the proof. □

Replacing q with p and replacing p with y^B in Proposition 1.179, we obtain the following using the Yoneda lemma.

Corollary 1.181. For any set B there is an adjunction

$$\mathbf{Set} \underset{p(B)}{\overset{Ay^B}{\underset{\longleftarrow}{\overset{\longrightarrow}{\Rightarrow}}}} \mathbf{Poly}$$

where the functors are labeled by where they send $p \in \mathbf{Poly}$ and $A \in \mathbf{Set}$.

Exercise 1.182. Prove Corollary 1.181 from Proposition 1.179. ◊

Proposition 1.183. The Yoneda embedding $A \mapsto y^A$ has a left adjoint

$$\mathbf{Set}^{\mathrm{op}} \underset{\Gamma}{\overset{y^-}{\underset{\longleftarrow}{\overset{\longrightarrow}{\Leftarrow}}}} \mathbf{Poly}$$

where $\Gamma(p) \coloneqq \prod_{i \in p(1)} p[i]$.

Proof. We have the following chain of natural isomorphisms:

$$\begin{aligned}
\mathbf{Set}(A, \Gamma(p)) &\cong \Big(\prod_{i \in p(1)} p[i] \Big)^A && \text{Definition of function} \\
&\cong \prod_{i \in p(1)} p[i]^A && \text{Definition of product} \\
&\cong \prod_{i \in p(1)} \sum_{j \in 1} p[i]^A && \text{Trivial sum} \\
&\cong \mathbf{Poly}(p, y^A). && Eq.\ (1.63)
\end{aligned}$$

□

Exercise 1.184. Show that $\Gamma(p) \cong [p, y](1)$ where $[-,-]$ is as in Proposition 1.163. ◊

Epi-mono factorization.

Proposition 1.185. Let $(f_1, f^\sharp)\colon p \to q$ be a morphism in **Poly**. It is a monomorphism iff the function $f_1\colon p(1) \to q(1)$ is a monomorphism in **Set** and, for each $i \in p(1)$ the function $f_i^\sharp\colon q[f_1(i)] \to p[i]$ is an epimorphism in **Set**.

Proof. $\Rightarrow$: Suppose that f is a monomorphism. Since $p \mapsto p(1)$ is a right adjoint (Theorem 1.175), it preserves monomorphisms. We need to show that for any $i \in p(1)$ the function $f_i^\sharp\colon q[f_1(i)] \to p[i]$ is an epimorphism in **Set**. Suppose given a set A and a pair of functions $g^\sharp, h^\sharp\colon p[i] \rightrightarrows A$ with $g^\sharp f_i^\sharp = h^\sharp f_i^\sharp$. They can be identified with morphisms $y^A \rightrightarrows p$ whose compositions with f are equal, hence $g = h$ by assumption, and hence $g^\sharp = h^\sharp$ as desired.
$\Leftarrow$: Suppose that f_1 is a monomorphism and that for each $i \in p(1)$ the function $f_i^\sharp$ is an epimorphism. Let r be a polynomial, $g, h\colon r \rightrightarrows p$ two morphisms, and suppose $fg = fh$. Then $f_1 g_1 = f_1 h_1$, which implies $g_1 = h_1$; we'll consider g_1 the default representation. We also have that $g_k^\sharp f_{g_1 k}^\sharp = h_k^\sharp f_{g_1 k}^\sharp$ for any $k \in r(1)$. But $f_{g_1 k}^\sharp$ is an epimorphism, so in fact $g_k^\sharp = h_k^\sharp$, as desired. □

Example 1.186. Choose a finite nonempty set k for $1 \leq k \in \mathbb{N}$, e.g. $\mathsf{k} = 12$. There is a monomorphism

$$(f, f^\sharp)\colon \mathsf{k}y^{\mathsf{k}} \to \mathbb{N}y^{\mathbb{N}}$$

such that the trajectory "going around and around the k-clock" comes from the usual counting trajectory Example 1.113 $\mathbb{N}y^{\mathbb{N}} \to y$.

On positions we have $f(i) = i$, for all natural numbers $1 \leq i \leq k$. On directions we have $f^\sharp(n) = n \mod k$.

Exercise 1.187. In Example 1.186 we gave a map $12y^{12} \to \mathbb{N}y^{\mathbb{N}}$. This allows us to turn any dynamical system with $\mathbb{N}$-many states into a dynamical system with 12 states, while keeping the same interface, say p.

Explain how the behavior of the new system $12y^{12} \to p$ would be seen to relate to the behavior of the old system $\mathbb{N}y^{\mathbb{N}} \to p$. ◊

Proposition 1.188. Let $(f_1, f^\sharp)\colon p \to q$ be a morphism in **Poly**. It is an epimorphism iff the function $f_1\colon p(1) \to q(1)$ is an epimorphism in **Set** and, for each $j \in q(1)$ the induced function

$$f_j^\flat\colon q[j] \to \prod_{\{i \in p(1) \mid f_1(i) = j\}} p[i]$$

from (1.79) is a monomorphism.

Proof. $\Rightarrow$: Suppose that f is an epimorphism. Since $p \mapsto p(1)$ is a left adjoint (Theorem 1.175), it preserves epimorphisms. We need to show that for any $j \in q(1)$ the function $f_j^\flat$ is a monomorphism in **Set**. Suppose given a set A and a pair of functions $g^\sharp, h^\sharp \colon A \rightrightarrows q[j]$ with $f_j^\flat g^\sharp = f_j^\flat h^\sharp$. They can be identified with morphisms $g, h \colon q \rightrightarrows y^A + 1$, which send the j-component to the first component, y^A, and send all other component to the second component, 1. It is easy to check that $fg = fh$, hence $g = h$, and hence $g^\sharp = h^\sharp$ as desired.
$\Leftarrow$: Suppose that f_1 is an epimorphism and that for each $j \in q(1)$ the function $f_j^\flat$ is a monomorphism. Let r be a polynomial, $g, h \colon q \rightrightarrows r$ two morphisms, and suppose $gf = hf$. Then $g_1 f_1 = h_1 f_1$, which implies $g_1 = h_1$; we'll consider g_1 the default representation. We also have that $f_i^\sharp g_{f\,i}^\sharp = f_i^\sharp h_{f\,i}^\sharp$ for any $i \in p(1)$. Then, in particular, for any $j \in q(1)$ the two composites

$$r[g_1 j] \underset{h_j^\sharp}{\overset{g_j^\sharp}{\rightrightarrows}} q[j] \xrightarrow{f_j^\flat} \prod_{\{i \in p(1) \mid f_1(i) = j\}} p[i]$$

are equal, which implies that $g_j^\sharp = h_j^\sharp$ as desired. □

Exercise 1.189. Suppose f is both a mono and an epi; it is an iso? (That is, is **Poly** *balanced*?) ◊

Exercise 1.190 (Epi-mono factorization). Suppose $p, q \in$ **Poly** are polynomials and $f = (f_1, f^\sharp) \colon p \to q$ is a morphism with notation as in Eq. (1.59).

1. Can every morphism in **Poly** be factored as an epic followed by a monic?
2. Is your factorization unique up to isomorphism? ◊

1.4.2 Limits, colimits, and cartesian closure

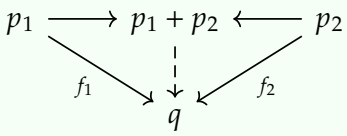
We will now see that **Poly** has all limits and colimits, and moreover it is cartesian closed.

Let's begin with something fairly simple: that + gives a coproduct in **Poly**. We could have said this elementary fact much earlier, but we were in a rush to get to the applications in Section 1.3.

Exercise 1.191. Show that the category **Poly** has finite coproducts as follows.

1. Show that 0 is an initial object in **Poly**, i.e. that for any polynomial p there is a unique morphism $0 \to p$.
2. Show that the standard sum of polynomials $p_1 + p_2$ is a coproduct of p_1 and p_2. That is, provide morphisms $p_1 \to p_1 + p_2 \leftarrow p_2$ and show that for any other q with

morphisms $p_1 \xrightarrow{f_1} q \xleftarrow{f_2} p_2$, there exists a unique morphism $p_1 + p_2 \to q$—shown dashed—making the following diagram commute



◊

The general case of coproducts and products is not much more difficult.

Proposition 1.192. The category **Poly** has coproducts and products.

Proof. Let A be a set and $p\colon A \to \mathbf{Poly}$ a collection of polynomials $(p_a)_{a\in A}$.Their coproduct is

$$\sum_{a\in A} p_a = \sum_{a\in A} \sum_{i\in p_a(1)} y^{p_a[i]} \cong \sum_{(a,i)\in\sum_{a\in A} p_a(1)} y^{p_a[i]} \tag{1.193}$$

which is manifestly a sum of representables, i.e. a polynomial, and which one can check satisfies the universal property of coproduct; see Exercise 1.195.

We claim that the product of this collection is

$$\prod_{a\in A} p_a = \prod_{a\in A} \sum_{i\in p_a(1)} y^{p_a[i]} \cong \sum_{i\colon \prod_{a\in A} p_a(1)} y^{\sum_{a\in A} p_a[i(a)]}. \tag{1.194}$$

Again, it is a sum of representables, so a polynomial, and one can check that it satisfies the universal property of product; see Exercise 1.195. □

Exercise 1.195. Finish the proof of Proposition 1.192 as follows.

1. Show that (1.193) satisfies the universal property for coproduct of polynomials.
2. Show that (1.194) satisfies the universal property for product of polynomials. ◊

Exercise 1.196. Let $A = 2$ and $p\colon A \to \mathbf{Poly}$ be given by $p_1 := y + 1$ and $p_2 := y + 2$. What is $\prod_{i\in 2} p[i]$ according to Eq. (1.194)? Does it check out? ◊

Proposition 1.197. The category **Poly** is completely distributive, i.e. for any set A, sets $(B_a)_{a\in A}$, and polynomials $(p_{(a,b),})_{a\in A, b\in B_a}$, there is an isomorphism

$$\sum_{b\colon \prod_{a\in A} B(a)} \prod_{a\in A} p_{(a,b(a)),} \cong \prod_{a\in A} \sum_{b\in B(a)} p_{(a,b),}$$

Proof. By the universal property of coproducts and products, to provide a morphism $\sum_{b\colon \prod_{a\in A} B(a)} \prod_{a\in A} p_{(a,b),} \to \prod_{a\in A} \sum_{b\in B(a)} p_{(a,b),}$, it suffices to fix an arbitrary $b_0 \in \prod_{a\in A} B(a)$ and $a_0 \in A$ and provide a morphism $\prod_{a\in A} p_{(a,b_0(a)),} \to \sum_{b\in B(a_0)} p_{(a_0,b),}$. We do so by

first projecting onto the a_0-factor $\prod_{a\in A} p_{(a,b_0(a)),} \to p_{(a_0,b_0(a_0)),}$ and then including into the $b_0(a_0)$-summand $p_{(a_0,b_0(a_0)),} \to \sum_{b\in B(a_0)} p_{(a_0,b),}$. This establishes the morphism.

To see that it is an isomorphism in **Poly**, i.e. a natural isomorphism of functors, it suffices to check it on components. So fix $X \in$ **Set**. The induced map

$$\sum_{b\colon \prod_{a\in A} B(a)} \prod_{a\in A} p_{(a,b(a)),} \cong \prod_{a\in A} \sum_{b\in B(a)} p_{(a,b),}$$

is exactly as in Proposition 1.74, which we proved is an isomorphism. This completes the proof. □

Exercise 1.198. How is the usual distributive law,

$$p(q+r) \cong pq + pr$$

for $p, q, r \in$ **Poly**, a special case of Proposition 1.197? ◊

Cartesian closure For any two polynomials q, r, define $r^q \in$ **Poly** by the following formula

$$r^q := \prod_{i\in q(1)} r \circ (y + q[j]) \tag{1.199}$$

where $\circ$ denotes composition.

Before proving that this really is an exponential in **Poly**, which we do in Theorem 1.202, we first get some practice with it.

Example 1.200. Let A be a set. We've been writing the polynomial Ay^0 simply as A and the polynomial y^1 simply as y, so it better be true that the there is an isomorphism

$$y^A \cong (y^1)^{(Ay^0)}$$

in order for the notation to be consistent. Luckily, this is true. Checking (1.199), we have

$$(y^1)^{Ay^0} = \prod_{a\in A} y \circ (0 + y) \cong y^A$$

Exercise 1.201. Compute the following exponentials in **Poly** using (1.199):

1. p^0 for an arbitrary $p \in$ **Poly**.
2. p^1 for an arbitrary $p \in$ **Poly**.
3. 1^p for an arbitrary $p \in$ **Poly**.
4. A^p for an arbitrary $p \in$ **Poly** and $A \in$ **Set**.
5. y^y.
6. y^{4y}.

7. $(y^A)^{y^B}$ for arbitrary sets $A, B \in \mathbf{Set}$. ◊

Theorem 1.202. The category **Poly** is Cartesian closed. That is, we have a natural isomorphism

$$\mathbf{Poly}(p, r^q) \cong \mathbf{Poly}(p \times q, r),$$

where r^q is the polynomial defined in (1.199).

Proof. We have the following chain of natural isomorphisms

$$\begin{aligned}
\mathbf{Poly}(p, r^q) &\cong \mathbf{Poly}\Big(p, \prod_{j \in q(1)} r \circ (y + q[j])\Big) \\
&\cong \prod_{j \in q(1)} \mathbf{Poly}(p, r \circ (y + q[j])) \\
&\cong \prod_{j \in q(1)} \prod_{i \in p(1)} \mathbf{Poly}(y^{p[i]}, r \circ (y + q[j])) \\
&\cong \prod_{i \in p(1)} \prod_{j \in q(1)} r \circ (p[i] + q[j]) \\
&\cong \prod_{i \in p(1)} \prod_{j \in q(1)} \sum_{k \in r(1)} (p[i] + q[j])^{r[k]} \\
&\cong \mathbf{Poly}(p \times q, r).
\end{aligned}$$

The last step uses Eq. (1.63) and Proposition 1.83. □

Exercise 1.203. Use Theorem 1.202 to show that for any polynomials p, q, there is a canonical evaluation map

$$\mathrm{eval}\colon q \times p^q \to p. \tag{1.204}$$

◊

Exercise 1.205. Using Eq. (1.199), show that the functor **Set** $\to$ **Poly** sends exponentials to exponentials. ◊

Theorem 1.206. The category **Poly** has all limits.

Proof. Recall that a category has all limits iff it has equalizers and products, so by Proposition 1.192 it suffices to show that **Poly** has equalizers.

Let $f_1, f_2\colon p \rightrightarrows q$ be two maps of polynomials. Let $E \to p(1) \rightrightarrows q(1)$ be an equalizer of the functions $f_1(1)$, and $f_2(1)$ in **Set**. For each $e \in E$, write $f(e) \coloneqq f_1(e) = f_2(e)$;

we have two polynomial morphisms $y^{p_e} \rightrightarrows y_{q[f(e)]}$, i.e. two functions $q[f(e)] \rightrightarrows p_e$. Defining $p'[e] \in \mathbf{Set}$ to be their coequalizer, we can define a polynomial p' as follows:

$$p' \coloneqq \sum_{e \in E} y^{p'[e]}$$

which comes equipped with a morphism $g\colon p' \to p$. One can check that it is an equalizer of f_1, f_2; see Exercise 1.207. □

Exercise 1.207. Complete the proof of Theorem 1.206 as follows:

1. We said that p' comes equipped with a morphism $g\colon p' \to p$; what is it?
2. Show that $g \fatsemi f_1 = g \fatsemi f_2$.
3. Show that g is an equalizer of the pair f_1, f_2. ◊

Exercise 1.208. Let p be any polynomial.

1. There is a canonical choice of morphism $\eta\colon p \to p(1)$; what is it?
2. Suppose given an element $i \in p(1)$, i.e. a function $1 \to p(1)$.
3. What is the pullback

$$\begin{array}{ccc} ? & \longrightarrow & p \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \eta} \\ 1 & \xrightarrow[i]{} & p(1) \end{array}$$

◊

Example 1.209 (Pullbacks in **Poly**). Given polynomials q, q', r and dependent lenses $q \xrightarrow{f} r \xleftarrow{f'} q'$, the pullback

$$\begin{array}{ccc} p & \longrightarrow & q' \\ \downarrow & \lrcorner & \downarrow{\scriptstyle f'} \\ q & \xrightarrow[f]{} & r \end{array}$$

is given as follows. The set of positions in p is the pullback of the positions of q and q' over those of r. On each p-position, say (i, i') with $f_1(i) = f'_1(i')$, we take the directions set of p in that position to be the pushout of the directions $q[i]$ and $q'[i']$ over $r[f_1(i)] = r[f'_1(i')]$:

$$\begin{array}{ccc} p(1) & \longrightarrow & q'(1) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle f'_1} \\ q(1) & \xrightarrow[f_1]{} & r(1) \end{array} \qquad \text{and} \qquad \begin{array}{ccc} p[(i,i')] & \longleftarrow & q'[i'] \\ \uparrow & \ulcorner & \uparrow{\scriptstyle (f')^\sharp_{i'}} \\ q[i] & \xleftarrow[f_i^\sharp]{} & r[f_1(i)] \end{array} \tag{1.210}$$

Exercise 1.211. Let $q \coloneqq y^2 + y$, $q' \coloneqq 2y^3 + y^2$, and $r \coloneqq y + 1$.

1. Choose morphisms $f : q \to r$ and $f' : q' \to r$ and write them down.
2. Find the pullback of your diagram. ◊

Arbitrary colimits will be much less useful to us than limits, so the following theorem is included only for completeness; the reader can feel free to skip it.

Theorem 1.212. The category **Poly** has all colimits.

Proof. Recall that a category has all colimits iff it has coequalizers and coproducts, so by Proposition 1.192 it suffices to show that **Poly** has coequalizers.

Let $f_1, f_2 : p \rightrightarrows q$ be two maps of polynomials. The pair of functions

$$f_1(1), f_2(1) : p(1) \rightrightarrows q(1)$$

define a graph $G : \boxed{\bullet \rightrightarrows \bullet} \to \mathbf{Set}$ whose set C of connected components is given by the coequalizer $g : q(1) \to C$ of $f_1(1)$ and $f_2(1)$. The coequalizer of f_1 and f_2 will turn out to be a C-indexed sum of representables, each of which is given by a limit of a diagram of representables from p and q, but expressing this limit, as we proceed to do, is a bit involved.

For each connected component $c \in C$, we have a connected subgraph $G_c \subseteq G$ with vertices $V_c \coloneqq g^{-1}(c)$ and edges $E_c \coloneqq f_1^{-1}(g^{-1}(c)) = f_2^{-1}(g^{-1}(c))$. Note that $E_c \subseteq p(1)$ and $V_c \subseteq q(1)$, so to each $e \in E_c$ (resp. to each $v \in V_c$) we have an associated representable $y^{p[e]}$ (resp. $y^{q[v]}$).

The category of elements $\int G_c$ is a bipartite graph with objects $V + E$ and with two sorts of morphisms, $e \to f_1(e)$ and $e \to f_2(e)$, associated to each $e \in E_c$; all non-identity arrows point from an object in E to an object in V. There is a functor $F : (\int G_c)^{\mathrm{op}} \to \mathbf{Set}$ sending every $v \mapsto q[v]$, every $e \mapsto p[e]$, and every morphism to a function between them, namely either $(f_1^\sharp)_e : q[f_1(e)] \to p[e]$ or $(f_2^\sharp)_e : q[f_2(e)] \to p[e]$. Define $q'[c] \in \mathbf{Set}$ to be the limit $q'[c] \coloneqq \lim F \in \mathbf{Set}$ of F.

We claim that $q' \coloneqq \sum_{c \in C} y^{q'[c]}$ is the coequalizer of f_1 and f_2. We leave the completion proof to the interested reader in Exercise 1.213. □

Exercise 1.213. Complete the proof of Theorem 1.212 as follows:

1. Provide a map $g : q \to q'$ and show that $f_1 \fatsemi g = f_2 \fatsemi g$.
2. Show that g is a coequalizer of the pair f_1, f_2. ◊

1.4.3 Monoidal ∗-bifibration over Set

We will see that the functor $p \mapsto p(1)$ has special properties making it what [**shulman2008framed**] refers to as a *monoidal ∗-bifibration*. This means that **Set** acts as a sort of remote controller on the category **Poly**, grabbing every polynomial by its positions and pushing or pulling it this way and that.

For example, suppose one has a set A and a function $f : A \to p(1)$. Then we get a new polynomial f^*p with positions A. The notation here agrees with that in **??**: it is given by a pullback

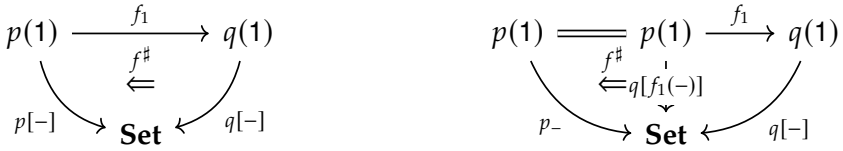
$$\begin{array}{ccc} f^*p & \xrightarrow{\text{cart}} & p \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \eta_p} \\ A & \xrightarrow[f]{} & p(1) \end{array} \tag{1.214}$$

Here η_p is the unit of the adjunction $\mathbf{Set} \underset{p(1)}{\overset{A}{\underset{\Leftarrow}{\overset{\longrightarrow}{\longleftarrow}}}} \mathbf{Poly}$. Since it is an isomorphismon positions and $p \mapsto p(1)$ is a right adjoint, the map $f^*p \to A$ is also an isomorphism on positions. Since on each position $a \in A$, the function $f^\sharp_a$ is an isomorphism on directions (both domain and codomain are the empty set), and since the a-position of f^*p is constructed by a pushout (see Example 1.209), each function $\text{cart}^\sharp_a : p[f(a)] \to (f^*p)[a]$ is an isomorphism too. We'll see this as part of a bigger picture in Proposition 1.227 and Theorem 1.228.

Definition 1.215 (Vertical, cartesian)**.** Let $f : p \to q$ be a morphism of polynomials. It is called *vertical* if $f_1 : p(1) \to q(1)$ is an isomorphism. It is called *cartesian* if, for each $i \in p(1)$ the function $f^\sharp_i : q[f(i)] \to p[i]$ is an isomorphism.

Proposition 1.216. Every morphism in **Poly** can be uniquely factored as a vertical morphism followed by a cartesian morphism.

Proof. Recall from Eq. (1.59) that a morphism in **Poly** can be written as to the left; we can thus rewrite it as to the right:



The object $\left(\begin{smallmatrix} q[f_1 i] \\ i \in p(1) \end{smallmatrix}\right)$ is clearly unique up to isomorphism. □

Proposition 1.217. Vertical morphisms satisfy 2/3: given $p \xrightarrow{f} q \xrightarrow{g} r$ with $h = g \circ f$, if any two of f, g, h are vertical, then so is the third.

If g is Cartesian then h is cartesian iff f is cartesian.

Exercise 1.218. Give an example of polynomials p, q, r and maps $p \xrightarrow{f} q \xrightarrow{g} r$ such that

f and $g \circ f$ are cartesian but g is not. ◊

Proposition 1.219. A morphism $f : p \to q$ is cartesian iff it is cartesian as a natural transformation, i.e. for any sets A, B and function $g : A \to B$, the naturality square

$$\begin{array}{ccc} p(A) & \xrightarrow{f_A} & q(A) \\ {\scriptstyle p(g)}\downarrow & \lrcorner & \downarrow{\scriptstyle q(g)} \\ p(B) & \xrightarrow[g_A]{} & q(B) \end{array}$$

is a pullback.

Exercise 1.220. Prove Proposition 1.219. ◊

Proposition 1.221. The monoidal structures $+, \times$, and $\otimes$ preserve cartesian morphisms.

Proof. Suppose that $f : p \to p'$ and $g : q \to q'$ are cartesian.

A position of $p + q$ is a position $i \in p(1)$ or a position $j \in q(1)$, and the map $(f + g)^\sharp$ at that position is either $f_i^\sharp$ or $g_j^\sharp$; either way it is an isomorphism, so $f + g$ is cartesian.

A position of $p \times q$ (resp. of $p \otimes q$) is a pair $(i, j) \in p(1) \times q(1)$. The morphism $(f \times g)^\sharp_{i,j}$ is $f_i^\sharp + g_j^\sharp$ (resp. $f_i^\sharp \times g_j^\sharp$) which is again an isomorphism if $f_i^\sharp$ and $g_j^\sharp$ are. Hence $f \times g$ (resp. $f \otimes g$) is cartesian, completing the proof. □

Proposition 1.222. Let $f : p \to q$ be a morphism and $g : q' \to q$ a cartesian morphism. Then the pullback g' of g along p

$$\begin{array}{ccc} p \times_q q' & \longrightarrow & q' \\ \downarrow & \lrcorner & \downarrow{\scriptstyle g} \\ p & \xrightarrow[f]{} & q \end{array}$$

is cartesian.

Proof. This follows from Example 1.209 since the pushout of an isomorphism is an isomorphism. □

Theorem 1.223 (Cartesian morphisms in **Poly** are exponentiable)**.** If $f : p \to q$ is cartesian then the functor $f^* : \mathbf{Poly}/q \to \mathbf{Poly}/p$ given by pulling back $q' \to q$ along f is a left adjoint:

$$\mathbf{Poly}/p \underset{f_*}{\overset{f^*}{\underset{\longrightarrow}{\overset{\longleftarrow}{\Rightarrow}}}} \mathbf{Poly}/q$$

Proof. Fix $e : p' \to p$ and $g : q' \to q$.

$$\begin{array}{ccc} p' & & q' \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle g} \\ p & \xrightarrow[f]{} & q. \end{array}$$

We need to define a functor $f_* : \mathbf{Poly}/p \to \mathbf{Poly}/q$ and prove the analogous isomorphism establishing it as right adjoint to f^*. We first establish some notation. Given a set Q and sets $(P'_i)_{i\in I}$, each equipped with a map $Q \to P'_i$, let $Q/\sum_{i\in I} P'_i$ denote the coproduct in $Q/\mathbf{Set}$, or equivalently the wide pushout of sets P'_i with apex Q. Then we have the following formula for f_*p', which we write in larger font for clarity:

$$f_*p' \coloneqq \sum_{j\in q(1)} \sum_{i' \in \prod_{i\in f_1^{-1}(j)} e_1^{-1}(i)} y^{q[j]/\sum_{i\in f_1^{-1}(j)} p'[i'(i)]}. \tag{1.224}$$

Again, $q[j]/\sum_{i\in f_1^{-1}(j)} p'[i'(i)]$ is the coproduct of the $p'[i'(i)]$, taken in $q[j]/\mathbf{Set}$. Since $p[i] \cong q[f(i)]$ for any $i \in p(1)$ by the cartesian assumption on f, we have the following chain of natural isomorphisms

$$\begin{aligned}
(\mathbf{Poly}/p)(f^*q', p') &\cong \prod_{i\in p(1)} \prod_{\{j'\in q'(1) \mid g_1(j')=f_1(i)\}} \sum_{\{i'\in p'(1) \mid e_1(i')=i\}} (p[i]/\mathbf{Set})(p'[i'], p[i] +_{q[f(i)]} q'[j']) \\
&\cong \prod_{i\in p(1)} \prod_{\{j'\in q'(1) \mid g_1(j')=f_1(i)\}} \sum_{\{i'\in p'(1) \mid e_1(i')=i\}} (q[f(i)]/\mathbf{Set})(p'[i'], q'[j']) \\
&\cong \prod_{j\in q(1)} \prod_{\{j'\in q'(1) \mid g_1(j')=j\}} \prod_{\{i\in p(1) \mid f_1(i)=j\}} \sum_{\{i'\in p'(1) \mid e_1(i')=i\}} (q[j]/\mathbf{Set})(p'[i'], q'[j']) \\
&\cong \prod_{j\in q(1)} \prod_{\{j'\in q'(1) \mid g_1(j')=j\}} \sum_{i'\in \prod_{i\in f_1^{-1}(j)} e_1^{-1}(i)} \prod_{i\in f_1^{-1}(j)} (q[j]/\mathbf{Set})(p'[i'(i)], q'[j']) \\
&\cong \prod_{j\in q(1)} \prod_{\{j'\in q'(1) \mid g_1(j')=j\}} \sum_{i'\in \prod_{i\in f_1^{-1}(j)} e_1^{-1}(i)} (q[j]/\mathbf{Set})\Big(\sum_{i\in f_1^{-1}(j)} p'[i'(i)], q'[j'] \Big) \\
&\cong (\mathbf{Poly}/q)(q', f_*p')
\end{aligned}$$

□

Example 1.225. Let $p \coloneqq 2y^2$, $q \coloneqq y^2 + y^0$, and $f : p \to q$ the unique cartesian morphism between them. Then for any $e : p' \to p$ over p, (1.224) provides the following description for the pushforward f_*p'. We use the isomorphisms $p(1) \cong 2$ and $q(1) \cong 2$ to talk about the positions of p and q.

Over the $j = 2$ position (which has $q[2] = 0$), we have $\prod_{i\in f_1^{-1}(2)e_1^{-1}(i)} \cong 1$ is an empty product, and $q[2]/\sum_{i\in f_1^{-1}(2)} p'[i'(i)]$ is an empty sum (in $q_2/\mathbf{Set}$), so we get $y^0 \cong 1$.

Over the $j = 1$ position (which has $q[1] = 2$), we have $\prod_{i'\in f_1^{-1}(1)e_1^{-1}(i)} \cong e_1^{-1}(1)\times e_1^{-1}(2)$. For each element $i' \in e_1^{-1}(1)\times e_1^{-1}(2)$, the sets $p'[i'(1)]$ and $p'[i'(2)]$ are each the codomain

of a map from $q[1] = p[1] = p[2] = 2$ coming from $e^\sharp$, and $q[1]/\sum_{i\in\{1,2\}} p'[i'(i)]$ is the pushout $p'[i'(1)] \leftarrow 2 \rightarrow p'[i'(2)]$. Let's call that pushout set $X_{i'}$. Then in sum we have

$$f_* p' \cong \left(\sum_{i' \in e_1^{-1}(1) \times e_2^{-1}(2)} y^{X_{i'}} \right) + 1.$$

Remark 1.226. The category **Poly** is not locally cartesian closed; for example the map $y \to 1$ is not exponentiable. Indeed, in the notation of Theorem 1.223, let $p := y$ and $q := 1$, and let $p' = q' := y^2$ with $e\colon y^2 \to y$ either projection. We'll show that the formula $\mathbf{Poly}/p(f^*q', p') \cong^? \mathbf{Poly}/q(q', f_*p')$ is impossible to satisfy.

We have $f^*(y^2) \cong y^3$ and hence

$$\mathbf{Poly}/p(f^*q', p') \cong \mathbf{Poly}/y(y^2, y^3) \cong 3.$$

There is no possibility for f_*p' because

$$\mathbf{Poly}/q(q', f_*p') \cong \mathbf{Poly}(y^2, f_*p') \cong 2^{\sum_{i \in f_*p'(1)} (f_*p')_i}$$

will always be a power of 2, and 3 is not a power of 2.

For any set A, let $A.\mathbf{Poly}$ denote the category whose objects are polynomials p equipped with an isomorphism $A \cong p(1)$, and whose morphisms are polynomial maps respecting the isomorphisms with A.

Proposition 1.227 (Base change)**.** For any function $f : A \to B$, pullback f^* along f induces a functor $B.\mathbf{Poly} \to A.\mathbf{Poly}$, which we also denote f^*.

Proof. This follows from (1.210) with $q := A$ and $r := B$, since pullback of an iso is an iso. □

Theorem 1.228. For any function $f : A \to B$, the pullback functor f^* has both a left and a right adjoint

$$A.\mathbf{Poly} \underset{\xrightarrow[f_*]{\Leftarrow}}{\overset{f_!}{\underset{\Rightarrow}{\longleftarrow}}} B.\mathbf{Poly} \qquad (1.229)$$

Moreover $\otimes$ preserves the op-Cartesian arrows, making this a monoidal $*$-bifibration in the sense of [**shulman2008framed**].

Proof. Let p be a polynomial with $p(1) \cong A$. Then the formula for $f_!p$ and f_*p are given as follows:

$$f_! p \cong \sum_{b \in B} y^{\left(\prod_{a \mapsto b} p[a]\right)} \qquad \text{and} \qquad f_* p \cong \sum_{b \in B} y^{\left(\sum_{a \mapsto b} p[a]\right)} \qquad (1.230)$$

It may at first be counterintuitive that the left adjoint $f_!$ involves a product and the right adjoint f_* involves a sum. The reason for this comes from Proposition 1.56, namely that **Poly** is equivalent to the Grothendieck construction applied to the functor $\mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ sending each set A to the category $(\mathbf{Set}/A)^{\mathrm{op}}$. The fact that functions $f : A \to B$ induces an adjoint triple between $\mathbf{Set}/A$ and $\mathbf{Set}/B$, and hence between $(\mathbf{Set}/A)^{\mathrm{op}}$ and $(\mathbf{Set}/B)^{\mathrm{op}}$ explains the variance in (1.230) and simultaneously establishes the adjoint triple (1.229).

The functor $p \mapsto p(1)$ is strong monoidal with respect to $\otimes$ and strict monoidal if we choose the lens construction as our model of **Poly**. By Proposition 1.221, the monoidal product $\otimes$ preserves cartesian morphisms; thus we will have established the desired monoidal $*$-bifibration in the sense of **[shulman2008framed]** as soon as we know that $\otimes$ preserves op-cartesian morphisms.

Given f and p as above, the op-cartesian morphism is the morphism $p \to f_! p$ obtained as the composite $p \to f^* f_! p \to f_! p$ where the first map is the unit of the $(f_!, f^*)$ adjunction and the second is the cartesian morphism for $f_! p$. On positions $p \to f_! p$ acts as f, and on directions it is given by projection.

If $f : p(1) \to B$ and $f' : p'(1) \to B'$ are functions then we have

$$\begin{aligned} f_!(p) \otimes f'_!(p') &\cong \sum_{b \in B} \sum_{b' \in B'} y^{\left(\prod_{a \mapsto b} p[a]\right) \times \left(\prod_{a' \mapsto b'} p'[a']\right)} \\ &\cong \sum_{(b,b') \in B \times B'} y^{\left(\prod_{(a,a') \mapsto (b,b')} p[a] \times p[b]\right)} \\ &\cong (f_! \otimes f'_!)(p \otimes p') \end{aligned}$$

and the op-cartesian morphisms are clearly preserved since projections in the second line match with projections in the first. □

1.5 Summary and further reading

...

Thanks to Joachim Kock for telling me about the derivative $\dot{p}$ of a polynomial and the relationship between $\dot{p}(1)$ and the total number of leaves of p.

See work by Gambino and Kock, Joyal, Paul Taylor, Michael Abbott and Neil Ghani (containers), ...

Chapter 2

A different category of categories

We have seen that the category of polynomial functors—sums of representables **Set** → **Set** and the natural transformations between them—has quite a bit of well-interoperating mathematical structure. Further, it is an expressive way to talk about dynamical systems that can change their interface and wiring pattern based on their internal states.

In this chapter we will discuss a monoidal structure on **Poly** that is quite easy from the mathematical point of view—it is simply composition—but which is again remarkable both in terms of its semantics and the phenomena that emerge mathematically.

In particular, we will see that the comonoids for the composition monoidal structure on **Poly** are precisely categories. However, the morphisms are different—they are often called *cofunctors*—and so we get a second category $\mathbf{Cat}^\sharp$ of categories and cofunctors. But the core groupoids of each—the groupoid of small categories and all functor isomorphisms between them, as well as the groupoid of small categories and all cofunctor isomorphisms between them—are isomorphic as groupoids. In other words, the following slogan is justified:

Polynomial comonads are precisely categories.

Cofunctors are not too familiar, but we will explain how to think of them in a variety of ways. We will see that whereas a functor $\mathcal{C} \to \mathcal{D}$ gives a kind of "picture" of $\mathcal{C}$ inside $\mathcal{D}$, a cofunctor $\mathcal{C} \nrightarrow \mathcal{D}$ gives a kind of $\mathcal{D}$-shaped "crystallization" of $\mathcal{C}$, one that is intuitively more geometric, more like creating neighborhoods. We will see in Chapter 3 that there is another kind of morphism between comonoids, namely the bimodules, that are perhaps more familiar: they are the so-called *parametric right adjoints*, or in perhaps more friendly terms, *data migration functors* between copresheaf categories.

The plan for this chapter is to first introduce what is perhaps the most interesting monoidal structure on **Poly**, namely the composition product; we do so in Section 2.1. We'll give a bunch of examples and ways to think about it in terms that relate to dynamical systems and our work so far. Then in Section 2.2 we'll discuss comonoids in **Poly** and explain why they are categories in down-to-earth, set-theoretic terms. We will also discuss the morphisms between them.

Finally in Section 2.3 we will discuss the cofree comonoid construction that takes any polynomial and returns a category. We will show how it relates to decision trees, as one may see in combinatorial game theory.

2.1 The composition product

In **??** we saw that the category **Poly** of polynomial functors—otherwise known as dependent lenses—is a very well-behaved category in which to think about dynamical systems of quite a general nature.

But we left one thing—what in some sense is the most interesting part of the story—out entirely. That thing is quite simple to state, and yet has profound consequences. Namely: polynomials can be composed:

$$y^2 \circ (y+1) = (y+1)^2 \cong y^2 + 2y + 1.$$

What could be simpler?[1]

It turns out that this operation, which we'll see soon is a monoidal product, has a lot to do with time. There is a strong sense—made precise in Proposition 2.2—in which the polynomial $p \circ q$ represents "starting at a position i in p, choosing a direction in $p[i]$, landing at a position j in q, choosing a direction in $q[j]$, and then landing... somewhere."

The composition product has many surprises up its sleeve, as we'll see. We've told many of them to you already in Section 1.1.5. We won't amass them all here; instead, we'll take you through the story step by step. But as a preview, this chapter will get us into decision trees, databases, and more dynamics, and it's all based on $\circ$.

As in Eq. (1.52), we'll continue to denote polynomials with the following notation

$$p \cong \sum_{i \in p(1)} y^{p[i]}, \tag{2.1}$$

and refer to $p(1)$ as the set of positions, and for each $i \in p(1)$ we'll refer to $p[i]$ as the set of direction at position i.

2.1.1 Defining the composition product

We begin with the definition of composition product.

Proposition 2.2. Suppose $p, q \in \mathbf{Poly}$ are polynomial functors $p, q \colon \mathbf{Set} \to \mathbf{Set}$. Then their composite $p \circ q$ is again a polynomial functor and we have the following isomorphisms

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in q(1)} \prod_{e \in q[j]} y.$$

[1] If you're thinking "What could be more boring?", don't forget that our job is to connect this to the material in **??**. If that payoff doesn't intrigue you, then this chapter is not for you.

Proof. We can rewrite Eq. (2.1) for p and q as

$$p \cong \sum_{i \in p(1)} \prod_{d \in p[i]} y \qquad \text{and} \qquad q \cong \sum_{j \in q(1)} \prod_{e \in q[j]} y.$$

For any set X we have $(p \circ q)(X) = p(q(X)) = p(\sum_j \prod_e X) = \sum_i \prod_d \sum_j \prod_e X$, so (2.4) is indeed the formula for their composite. To see this is a polynomial, we use Proposition 1.197, which says we can rewrite the $\prod \sigma$ in (2.4) as a $\sigma \prod$. The result

$$p \circ q \cong \sum_{i \in p(1)} \sum_{j_i \colon p[i] \to q(1)} y^{\sum_{d \in p[i]} q[j_i(d)]}, \tag{2.3}$$

(written slightly bigger for clarity) is clearly a polynomial. □

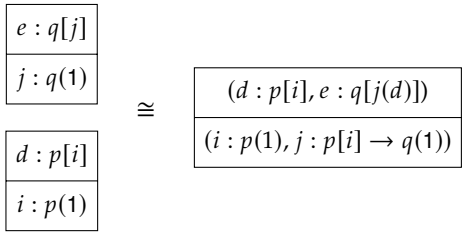
The composition of polynomials will be extremely important in the story that follows. However, we only sometimes think of it as composition; more often we think of it as a certain operation on arenas, or collections of corollas. Because we may wish to use $\circ$ to denote composition in arbitrary categories, we use a special symbol for polynomial composition namely

$$p \triangleleft q \coloneqq p \circ q.$$

The symbol $\triangleleft$ looks a bit like the composition symbol, in that it is an open shape, and when handwriting it fast, it's ok if it morphs into a $\circ$, but we'll soon see that it is quite evocative in terms of trees, and again it leaves $\circ$ for other uses.

We repeat the important formulas from Proposition 2.2 in the new notation:

$$p \triangleleft q \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in q(1)} \prod_{e \in q[j]} y. \tag{2.4}$$



Exercise 2.5. Let's consider (2.3) piece by piece, with concrete polynomials $p \coloneqq y^2 + y^1$ and $q \coloneqq y^3 + 1$.

1. What is q^2?
2. What is $y^2 \triangleleft q$?
3. What is $y^1 \triangleleft q$?
4. What is $(y^2 + y) \triangleleft q$? This is what $p \triangleleft q$ "should be".
5. How many functions $j_1 \colon p[1] \to q(1)$ are there?
6. For each function j_1 as above, what is $\sum_{d \in p[1]} q[j_1(d)]$?
7. How many functions $j_2 \colon p[2] \to q(2)$ are there?
8. For each function j_2 as above, what is $\sum_{d \in p[2]} q[j_2(d)]$?

9. Write out $\sum_{i \in p(1)} \sum_{j_i \colon p[i] \to q(1)} y^{\sum_{d \in p[i]} q[j_i(d)]}$.
10. Does the result agree with what $p \triangleleft q$ should be? ◊

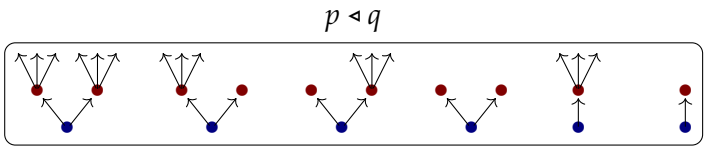
Exercise 2.6.

1. If p and q are representable, show that $p \triangleleft q$ is too. Give a formula for it.
2. If p and q are linear, show that $p \triangleleft q$ is too. Give a formula for it.
3. If p and q are constant, show that $p \triangleleft q$ is too. Give a formula for it. ◊

In terms of corollas, composition product $p \triangleleft q$ is given by glueing q-corollas onto the tips of p-corollas in every possible way. Let's say $p := y^2 + y$ and $q := y^3 + 1$, which as in Eq. (1.28) we draw as follows

(2.7)

Then their composite $p \triangleleft q$ would be drawn like so:



It has six positions; the first has six directions, the second, third, and fifth have three directions, and the fourth and sixth have no directions. In total, we read off that $p \triangleleft q$ is isomorphic to $y^6 + 3y^3 + 2$.

Exercise 2.8. Use p, q as in Eq. (2.7) and $r := y^2 + 1$ in the following.

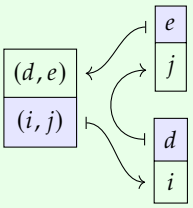
1. Draw $q \triangleleft p$.
2. Draw $p \triangleleft p$.
3. Draw $p \triangleleft p \triangleleft 1$.
4. Draw $r \triangleleft r$.
5. Draw $r \triangleleft r \triangleleft r$. ◊

Exercise 2.9. Let A and B be arbitrary sets, and let p be an arbitrary polynomial. Which of the following isomorphisms exist?

1. $(Ay) \otimes (By) \cong (Ay) \triangleleft (By)$?
2. $y^A \otimes y^B \cong y^A \triangleleft y^B$?
3. $A \otimes B \cong A \triangleleft B$?
4. $Ay \otimes p \cong Ay \triangleleft p$?
5. $y^A \otimes p \cong y^A \triangleleft p$?
6. $p \otimes Ay \cong p \triangleleft Ay$?

7. $p \otimes y^A \cong p \triangleleft y^A$? ◊

Example 2.10. For any p and q there is an interesting map $o_{p,q} : p \otimes q \to p \triangleleft q$ that orders the operation. It looks like this:



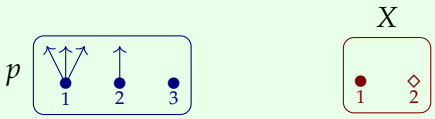
In other words, $p \triangleleft q$ is allowed to have j depend on d, whereas $p \otimes q$ is not; the map is in some sense the inclusion of the order-independent part. And of course we can flip the order using the symmetry $q \otimes p \cong p \otimes q$. This is, we just as well have a map $p \otimes q \to q \triangleleft p$.

Both $\otimes$ and $\triangleleft$ have the same monoidal unit, the identity functor y, and the identity is the unique map $y \to y$. The maps $o_{p,q}$ should commute with associators and unitors.

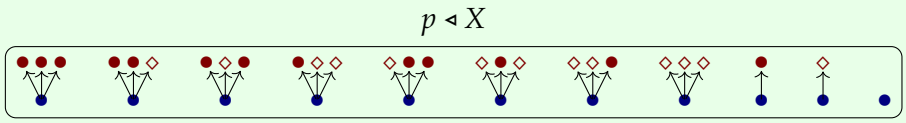
This can be used in the following way. Maps $p \to q \triangleleft r$ into composites are fairly easy to diagram and understand, whereas maps $q \triangleleft r \to p$ are not so easy to think about. However, given such a map, one may always compose it with $o_{q,r}$ to obtain a map $q \otimes r \to p$; this is quite a bit simpler to think about, more like a wiring diagram.

Example 2.11. For any set X and polynomial p, we can take $p(X) \in \mathbf{Set}$; indeed $p : \mathbf{Set} \to \mathbf{Set}$ is a functor! In particular, by this point you've seen us write $p(1)$ hundreds of times. But we've also seen that X is itself a polynomial, namely a constant one.

It's not hard to see that $p(X) \cong p \triangleleft X$. Here's a picture, where $p := y^3 + y + 1$ and $X := 2$.



Let's see how $(y^3 + y + 1) \triangleleft 2$ looks.



It has 11 positions and no open leaves, which means it's a set (constant polynomial), namely $p \triangleleft X \cong 11$.

We could also draw $X \triangleleft p$, since both are perfectly valid polynomials. Here it is:

$X \triangleleft p$

Each of the open leaves in X—of which there are none—is filled with a corolla from p.

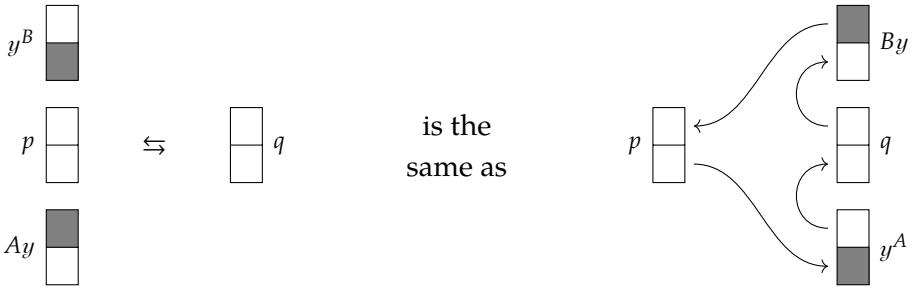
Exercise 2.12.

1. Choose a polynomial p and draw $p \triangleleft 1$ in the style of Example 2.11.
2. Is it true that if X is a set (considered as a constant polynomial) and p is any polynomial, then $X \triangleleft p \cong X$?
3. Is it true that if X is a set and p is a polynomial then $p \triangleleft X \cong p(X)$, where $p(X)$ is the set given by applying p as a functor to X? ◊

Proposition 2.13. For all sets A, B, we have the following adjunction:

$$\mathbf{Poly}\left(Ay \triangleleft p \triangleleft y^B, q\right) \cong \mathbf{Poly}\left(p, y^A \triangleleft q \triangleleft By\right)$$

Moreover, this isomorphism is natural in $A \in \mathbf{Set}^{\mathrm{op}}$ and $B \in \mathbf{Set}$.



Do you see how polyboxes with a black (one-element) part can flip upside-down to go to the other side?

Proof. We prove this in two pieces: that

$$\mathbf{Poly}\left(Ay \triangleleft p, q\right) \cong \mathbf{Poly}\left(p, y^A \triangleleft q\right) \tag{2.14}$$

and that

$$\mathbf{Poly}\left(p \triangleleft y^B, q\right) \cong \mathbf{Poly}\left(p, q \triangleleft By\right) \tag{2.15}$$

For Eq. (2.14), we have that $Ay \triangleleft p \cong Ap$, an A-fold coproduct of p. Similarly, $y^A \triangleleft q \cong q^A$, an A-fold product of q. So this follows from the corresponding universal properties.

For Eq. (2.15), we first write out the two sets by hand. To give a map from $p \triangleleft y^B$ to q, we must provide for every $i \in p(1)$ an element $j \in q(1)$ and a function $q[j] \to B \times p[i]$. Then to give a map from p to $q \triangleleft By$, we must provide for every $i \in p(1)$ an element $j \in q(1)$ and for every $n \in q[j]$, an element of B and an element of $p[i]$. These are clearly isomorphic. □

Exercise 2.16. Let $A, B \in \mathbf{Set}$ be sets, and let $p \in \mathbf{Poly}$ be a polynomial. Is it true that the morphisms $Ay^B \to p$ can be identified with the morphisms $A \to p \triangleleft B$, i.e. that there is a bijection:

$$\mathbf{Poly}(Ay^B, p) \cong^? \mathbf{Poly}(A, p \triangleleft B) \tag{2.17}$$

If so, why? If not, give a counterexample. ◊

Exercise 2.18. For any $p \in \mathbf{Poly}$ there are natural isomorphisms $p \cong p \triangleleft y$ and $p \cong y \triangleleft p$.

1. Thinking of polynomials as functors $\mathbf{Set} \to \mathbf{Set}$, what functor does y represent?
2. Why is $p \cong y$ isomorphic to p?
3. In terms of tree pictures, draw $y \triangleleft p$ and $p \triangleleft y$, and explain pictorially how to see the isomorphisms $y \triangleleft p \cong p \cong p \triangleleft y$. ◊

2.1.2 Monoidal structure $(\mathbf{Poly}, \triangleleft, y)$

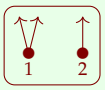
The technical claim is that $\triangleleft$ is a monoidal product, which means that it's well-behaved, in particular it's functorial, associative, and unital. In fact, all of this comes from general theory: for any category $\mathcal{C}$ the category whose objects are functors $\mathcal{C} \to \mathcal{C}$ and whose morphisms are natural transformations is a monoidal category. For us $\mathcal{C} := \mathbf{Set}$, and we are only using polynomial functors, not all functors, so there is a tiny bit to do, but it's accomplished by **??** and the fact that the identity functor $\mathbf{Set} \to \mathbf{Set}$ is a polynomial (it's y).

However, even though the formal theory of functors and natural transformations knocks the monoidality of $\triangleleft$ out of the park, it is still useful to discuss how it acts on morphisms in terms of positions and directions.

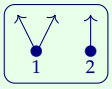
For any $f : p \to p'$ and $g : q \to q'$, we want to define a morphism $(f \triangleleft g) : (p \triangleleft q) \to (p' \triangleleft q')$. This is actually quite an impressive operation! It threads back and forth in a fascinating way.

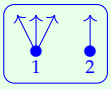
Recall from Example 1.66 that we can think of $f = (f_1, f^\sharp)$ as a way to delegate decisions from p to p'. Every decision (corolla / position) $i \in p(1)$ is assigned a decision $f_1(i) \in p'(1)$. Then every option $d \in p'[f_1(i)]$ there is passed back to an option $f^\sharp(d) \in p[i]$. Let's start with an example and then give the general method.

Example 2.19. Let's take $p := y^2 + y$, $q := y^2 + y$, $p' := y^3 + y$, and $q' := y + 1$.

$q =$ 

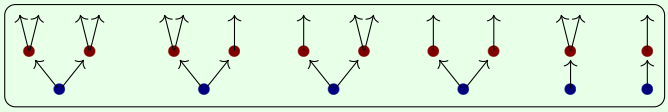
$q' =$ 

$p =$ 

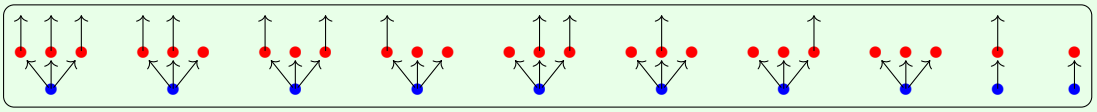
$p' =$ 

For any way to delegate from p to p' and q to q', we're supposed to give a way to delegate from $(p \triangleleft q)$ to $(p' \triangleleft q')$. Let's draw $p \triangleleft q$ and $p' \triangleleft q'$.

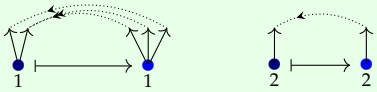
$p \triangleleft q$

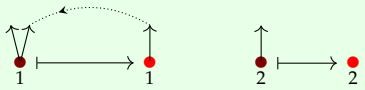


$p' \triangleleft q'$



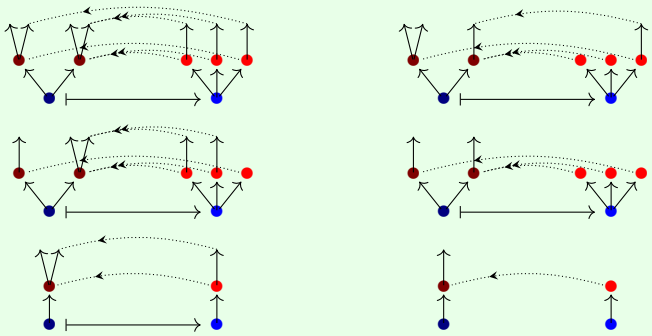
Ok, now suppose someone gives us delegations (morphisms / dependent lenses) $f : p \to p'$ and $g : q \to q'$. Let's just pick something relatively at random

$f : p \to p'$ 

$g : q \to q'$ 

Then we can form the induced delegation (morphism / dependent lens) $f \triangleleft g : (p \triangleleft q) \to (p' \triangleleft q')$ as follows. For each two-level tree (position in $p \triangleleft q$), we begin by using f to send the p-corolla on the bottom to a p'-corolla. The second-level nodes (from q') have not been chosen yet, but each of the p'-directions is passed back to a p-direction via $f^\sharp$. Now we use g to send the q-corolla at the second level to a q' corolla (this part is not shown in the diagram below because it would add clutter). Again each of the q'-directions is passed back to a q direction via $g^\sharp$.

Our six pictures below leave out the fact that the red corollas on the right are selected according to g; hopefully the reader can put it together for themselves.



Again, we're not making up these rules; it's a tree representation of how natural transformations f and g compose to form $f \triangleleft g$.

Exercise 2.20. With p, q, p', q' and f, g as in Example 2.19, draw $g \triangleleft f : (q \triangleleft p) \to (q' \triangleleft p')$ in terms of trees as in the example. ◊

Exercise 2.21. Suppose p, q, and r are polynomials and you're given arbitrary morphisms $f : q \to p \triangleleft q$ and $g : q \to q \triangleleft r$. Does the following diagram necessarily commute?

$$\begin{array}{ccc} q & \xrightarrow{g} & q \triangleleft r \\ {\scriptstyle f}\downarrow & ? & \downarrow{\scriptstyle f \triangleleft r} \\ p \triangleleft q & \xrightarrow[p \triangleleft g]{} & p \triangleleft q \triangleleft r \end{array}$$

That is, do we have $(p \triangleleft g) \circ f =^{?} (f \triangleleft r) \circ g$? ◊

Pronouncing polynomial composites. We want to be able to pronounce polynomials like p and q in some way, which is intuitive and which lends itself to pronouncing composites like $p \triangleleft q$ or $p \triangleleft p \triangleleft q \triangleleft p$. We pronounce the polynomial

$$p = \sum_{i \in p(1)} \prod_{d \in p[i]} y$$

as "a choice of p-position i and, for every direction $d \in p[i]$ there, a future." Other than the word "future" in place of y, this is just pronouncing dependent sums and products. By saying "a future", we indicate that y is a functor: for any set X one could put in its place, we'll get an element of that X. We know we're getting an element of something, we just don't yet know what.

To pronounce composites of polynomials $p \triangleleft q$, we pronounce almost all of p, except we replace "future" with q. More precisely, to pronounce $p \triangleleft q$, which has the formula

$$p \triangleleft q \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in q(1)} \prod_{e \in q[j]} y,$$

we would say "a choice of position $i \in p(1)$ and, for every direction $d \in p[i]$ there, a choice of position $j \in q(1)$ and, for every direction $e \in q[j]$ there, a future.

Exercise 2.22.

1. Let p be an arbitrary polynomial. Write out the English pronunciation of $p \triangleleft p \triangleleft p$.
2. Pronouncing the unique element of 1 as "completion", write out the pronunciation of $p \triangleleft p \triangleleft 1$.
3. Pronouncing $\prod_{d \in \varnothing} y$ as "with no directions to travel, a complete dissociation from any purported future", write out the pronunciation of $p \triangleleft p \triangleleft y^0$.
4. With the "dissociation" language, pronounce $p \triangleleft 1 \triangleleft p$, and see if it makes sense with the fact that $p \triangleleft 1 \triangleleft p \cong p \triangleleft 1$. ◊

For any $n \in \mathbb{N}$, let $p^{\triangleleft n}$ denote the n-fold $\triangleleft$ power of p, e.g. $p^{\triangleleft 3} := p \triangleleft p \triangleleft p$. In particular, $p^{\triangleleft 1} := p$ and $p^{\triangleleft 0} := y$. We might think of $p^{\triangleleft n}$ in terms of length-n strategies, in the sense of game theory, except that the opponent is somehow abstract, having no positions of its own. That is, we pronounce $p^{\triangleleft n}$

Exercise 2.23. Let $p, q \in \mathbf{Poly}$ be polynomials and $n \in \mathbb{N}$; say $n \geq 1$. Pronounce $(p \triangleleft q)^{\triangleleft(n+2)}$, using the exact phrase "and so on, n times, ending with". ◊

2.1.3 Working with composites

We need a way of talking about maps to composites.[2] The set of morphisms $p \to q_1 \triangleleft q_2 \triangleleft \cdots \triangleleft q_k$ has the following form:

$$\mathbf{Poly}(p,\ q_1 \triangleleft q_2 \triangleleft \cdots \triangleleft q_k) \cong \prod_{i \in p(1)} \sum_{j_1 \in q_1(1)} \prod_{e_1 \in q_1[j_1]} \sum_{j_2 \in q_2(1)} \cdots \prod_{e_k \in q_k[j_{k-1}]} \sum_{d \in p[i]} 1$$

We can use this to generalize our notation in the case $k = 1$, i.e. for morphisms $p \to q$. That is we denoted such a morphism by $\binom{f^\sharp}{f_1}$, where $f_1 \colon p(1) \to q(1)$ and $f_i^\sharp \colon q[f_1(i)] \to p[i]$. We generalize this to the k-ary composite case as

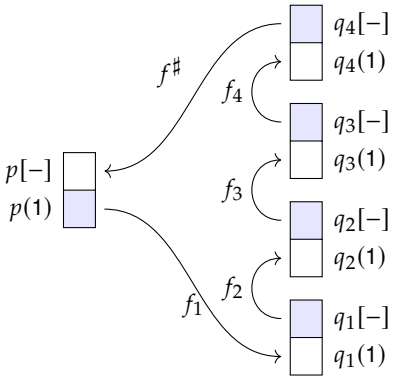
$$(f_1, f_2, \ldots, f_k, f^\sharp) \colon p \longrightarrow q_1 \triangleleft q_2 \triangleleft \cdots \triangleleft q_k, \tag{2.24}$$

[2] It would be nice to also have a nice way of talking about maps *out of* composites; however that is more difficult.

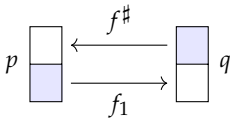
where

$$\begin{aligned}
&f_1 : p(1) \to q_1(1),\\
&f_2 : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to q_2(1),\\
&f_3 : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to (e_2 \in q_2[f_2(i,e_1)]) \to q_3(1),\\
&f_k : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to \cdots \to (e_{k-1} \in q_{k-1}[f_{k-1}(i,e_1,\ldots,e_{k-2})]) \to q_k(1),\\
&f^\sharp : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to \cdots \to (e_k \in q_k[f_k(i,e_1,\ldots,e_{k-1})]) \to p[i]
\end{aligned} \tag{2.25}$$

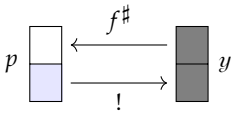
Here's a picture for the $k = 4$ case:



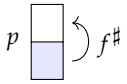
A few considerations might make (2.25) less scary. First of all, we're usually interested in the cases $k = 0, 1, 2$. The case $k = 1$ is just our $(f_1, f^\sharp)$ notation



The case $k = 0$ says that a map $p \to y$ is just a function $f^\sharp \in \prod_{i \in p(1)} p[i]$

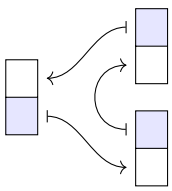


which we can rewrite simply as



(2.26)

The case $k = 2$ looks like this



(2.27)

and is considered in Example 2.28. But before we get there, let's think about (2.25) in terms of delegating decisions by "due process".

Suppose we're p and someone gives us a decision $i \in p(1)$ to make: we're supposed to pick an element of $p[i]$. Luckily, by virtue of our morphism $f : p \to q_1 \triangleleft \cdots \triangleleft q_k$, consisting of steps $f_1, \ldots, f_k$ and an interpretation $f^\sharp$, we have a process to follow by which we can make the decision. Our first step is to ask q_1 to make decision $f_1(i)$. It dutifully chooses some option, say $e_1 \in q_1[f_1(i)]$. We know exactly what to do: our second step is to ask q_2 to make decision $f_2(i, e_1)$. It dutifully chooses some option, say $e_2 \in q_2[f_2(i, e_1)]$. We continue with the plan through step k, at which point we ask q_k to make decision $f_k(i, e_1, \ldots, e_{k-1})$. It dutifully chooses some option, say e_k, which we then interpret via our passback function $f^\sharp$ to obtain the desired decision $f^\sharp(i, e_1, \ldots, e_k) \in p[i]$.

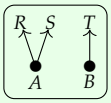
A morphism $p \to q_1 \triangleleft \cdots \triangleleft q_k$ is a multi-step policy for p to make decisions by asking for decisions from q_1 then q_2, etc., all the way until q_k, and interpreting the results.

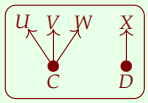
Example 2.28 (Maps $p \to q \triangleleft r$). By (2.25), a morphism $p \to q \triangleleft r$ can be specified by a tuple $(f_1, f_2, f^\sharp)$, where $f_1 : p(1) \to q(1)$ and $f_2, f^\sharp$ are a little more involved because they are dependent functions.

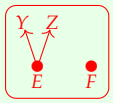
The dependent function f_2 takes as input a pair (i, d) where $i \in p(1)$ and $d \in q[f_1(i)]$, and it outputs an element of $r(1)$.

The dependent function $f^\sharp$ takes as input a tuple (i, d, e), where i, d are as above and $e \in r[f_2(i, d)]$, and it outputs an element of $p[i]$.

For example, let $p := \{A\}y^{\{R,S\}} + By^{\{T\}}$, $q := \{C\}y^{\{U,V,W\}} + \{D\}y^{\{X\}}$, and $r := \{E\}y^{\{Y,Z\}} + \{F\}$.

$p =$ 

$q =$ 

$r =$ 

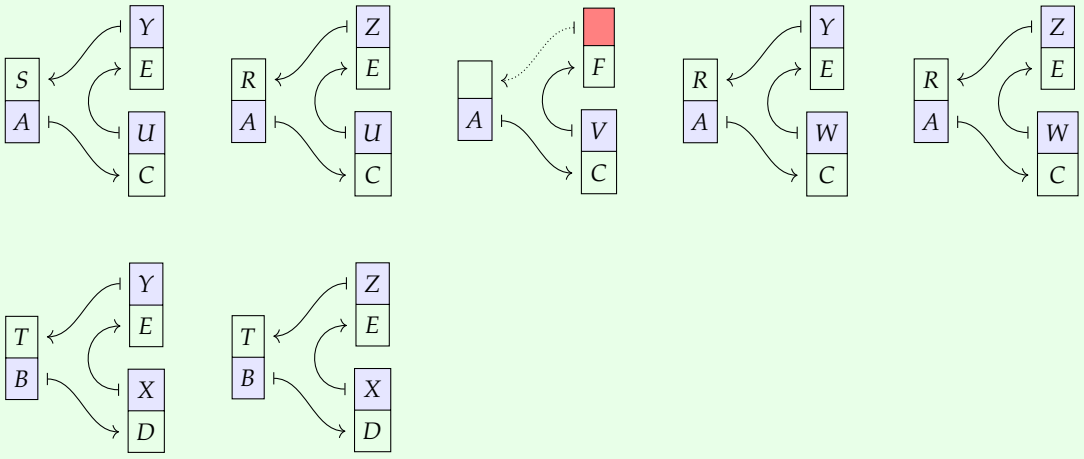
Here is a picture of a map $p \to q \triangleleft r$:

If we write it as $(f_1, f_2, f^\sharp) : p \to q \triangleleft r$ then we have

$$f_1(A) = C, \quad f_1(B) = D,$$
$$f_2(A, U) = E, \quad f_2(A, V) = F, \quad f_2(A, W) = E,$$
$$f_2(B, X) = E,$$

$$f^\sharp(A,U,Y) = S, \quad f^\sharp(A,U,Z) = R, \quad f^\sharp(A,W,Y) = R, \quad f^\sharp(A,W,Z) = R$$
$$f^\sharp(B,X,Y) = T, \quad f^\sharp(B,X,Z) = T$$

Box-notation uses a slightly different organization to represent the same data:



Example 2.29 (Using (2.24) to denote positions and directions in a composite). Suppose given polynomials $p_1, \ldots, p_k$. Recall from Exercise 1.178 that a position in their composite as a map

$$i \colon y^1 \to p_1 \triangleleft \cdots \triangleleft p_k.$$

We can denote i in the notation (2.24) as $i = (i_1, \ldots, i_k)$, forgoing the input to i_1 because it is always $1 \in 1$ and also forgoing $f^\sharp$ because it is always the unique map to 1. Then in this notation

$$i_1 \in p_1(1), \quad i_2 \colon p_1[i_1] \to p_2(1), \quad i_3 \colon (d_1 \in p_1[i_1]) \to (d_2 \in p_2[i_2(d_1)]) \to p_3(1),$$
$$i_k \colon (d_1 \in p_1[i_1]) \to (d_2 \in p_2[i_2(d_1)]) \to \cdots (d_{k-1} \in p_{k-1}[i_{k-1}(d_1, \ldots, d_{k-2})]) \to p_k(1)$$

So for example to give a position in $p \triangleleft q \triangleleft r$ we need

$$i \in p(1), \quad j \colon p[i] \to q(1), \quad k : (d \in p[i]) \to (e \in q[j(d)]) \to r(1).$$

The direction-set of $p_1 \triangleleft \cdots \triangleleft p_k$ at position $(i_1, \ldots, i_k)$ is

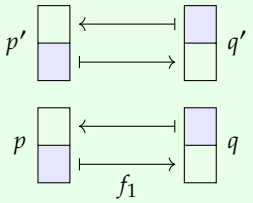
$$(p_1 \triangleleft \cdots \triangleleft p_k)[(i_1, \ldots, i_k)] \cong \sum_{d_1 \in p_1[i_1]} \sum_{d_2 \in p_2[i_2(d_1)]} \cdots \sum_{d_k \in p_k[i_k(d_1, \ldots, d_{k-1})]} 1$$

So for example given a position $(i, j, k) \in p \triangleleft q \triangleleft r$, a direction there consists of a tuple (d, e, f) where $d \in p[i]$, $e \in q[j(d)]$ and $f \in r[k(d, e)]$.

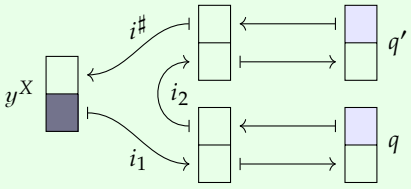
Exercise 2.30. Suppose $A_1, \ldots, A_k$ are sets and $p_i \coloneqq A_i y$ for each i. Use the notation of Example 2.29 to give the set of positions in $p \coloneqq p_1 \triangleleft \cdots \triangleleft p_k$. ◊

What if I give you a two-step decision to make of shape $p \triangleleft q$: I'll give you a position in p, you choose a direction there, and then based on your answer I'll give you a position in q, and you choose a direction there too. Now if each of p and q knew how to delegate its decisions to its partner, say $p \to p'$ and $q \to q'$, you should be able to delegate your two-step decision in $p \triangleleft q$ to a two-step decision by the partners $p' \triangleleft q'$. Here's how this looks in box-notation

Example 2.31 ($\triangleleft$ on morphisms). Given maps $f\colon p \to q$ and $f'\colon p' \to q'$, the corresponding map $(f \triangleleft f')\colon (p \triangleleft p') \to (q \triangleleft q')$ looks quite simple—even sterile—in box notation:



But it gets animated when someone chooses a map from an arbitrary representable (or anything else); to do so is to choose a bunch of arrows as to the left:



One can now visualize the information flow through this sequence of delegations.

Exercise 2.32. Draw a picture analogous to (**??**) for the map $(f \triangleleft g \triangleleft h)\colon (p \triangleleft q \triangleleft r) \to (p' \triangleleft q' \triangleleft r')$, given $f = (f_1, f^\sharp)\colon p \to p'$, $g = (g_1, g^\sharp)\colon q \to q'$, and $h = (h_1, h^\sharp)\colon r \to r'$. Show what happens when one adds a map $y^X \to p \triangleleft q \triangleleft r$ from a representable. ◊

2.1.4 Mathematical aspects of $\triangleleft$

We now want to get at more subtle aspects of $p \triangleleft q$. We begin with the following.

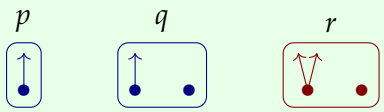
Proposition 2.33 (Left distributivity of $\triangleleft$)**.** For any polynomial r, the post-compose-with-r functor $(- \triangleleft r)\colon \mathbf{Poly} \to \mathbf{Poly}$ commutes—up to natural isomorphism—with addition and multiplication:

$$(p + q) \triangleleft r \cong (p \triangleleft r) + (q \triangleleft r) \qquad \text{and} \qquad pq \triangleleft r \cong (p \triangleleft r)(q \triangleleft r).$$

Proof. Formally, this just comes down to the fact that coproducts and products of functors **Set** $\to$ **Set** are computed pointwise and **Poly** is a full subcategory of $\mathbf{Set}^{\mathbf{Set}}$. One could instead give a proof in terms of $\sum$'s and $\prod$'s; this is done in Exercise 2.34. □

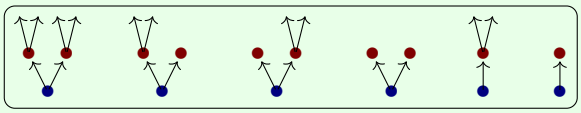
Exercise 2.34. Prove Proposition 2.33 in terms of the formula for $\triangleleft$ given in Proposition 2.2. ◊

Example 2.35 (Picturing the left distributivity of $\triangleleft$ over $\times$). We want an intuitive understanding of this left-distributivity. Let $p := y$, $q := y + 1$, and $r := y^2 + 1$, as shown here:



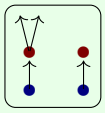
Then $pq \cong y^2 + y$ and we can draw $pq \triangleleft r$ as follows:

$$(p \times q) \triangleleft r \cong$$

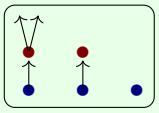


Or we can compute $p \triangleleft r$ and $q \triangleleft r$ seperately:

$$p \triangleleft r \cong$$

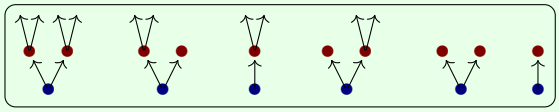


$$q \triangleleft r \cong$$



and multiply them together by taking each tree from $p \triangleleft r$ and pairing it with each tree from $q \triangleleft r$:

$$(p \triangleleft q) \times (p \triangleleft r) \cong$$



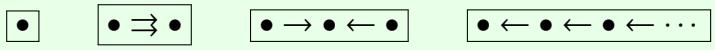
Exercise 2.36. Follow Example 2.35 with $+$ in place of $\times$: use pictures to give an intuitive understanding of the left-distributivity $(p + q) \triangleleft r \cong (p \triangleleft r) + (q \triangleleft r)$. ◊

Exercise 2.37. Show that the distributivities of Example 2.35 and Exercise 2.36 do not hold on the other side:

1. Find polynomials p, q, r such that $p \triangleleft (qr) \ncong (p \triangleleft q)(p \triangleleft r)$.
2. Find polynomials p, q, r such that $p \triangleleft (q + r) \ncong (p \triangleleft q) + (p \triangleleft r)$. ◊

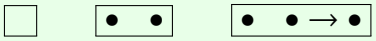
A connected limit is one whose indexing category J is (nonempty and) connected. That is, J has at least one object and any two objects are connected by a finite zigzag of arrows.

Example 2.38. The following categories are connected:



In particular, equalizers, pullbacks, and directed limits are examples of connected limits.

The following categories are *not connected*:



In particular, terminal objects and products are *not* examples of connected limits.

Theorem 2.39 (Preservation of connected limits)**.** The operation $\triangleleft$ commutes with connected limits in both variables. That is, if J is a connected category, $p\colon J \to \mathbf{Poly}$ is a functor, and $q \in \mathbf{Poly}$ is a polynomial, then there are natural isomorphisms

$$\left(\lim_{j\in J} p_j\right) \triangleleft q \cong \lim_{j\in J}(p_j \triangleleft q) \qquad \text{and} \qquad q \triangleleft \left(\lim_{j\in J} p_j\right) \cong \lim_{j\in J}(q \triangleleft p_j)$$

Sketch of proof. The claim for the left variable follows as in the proof of Proposition 2.33: limits of functors $\mathbf{Set} \to \mathbf{Set}$ are computed pointwise and **Poly** is a full subcategory of $\mathbf{Set}^{\mathbf{Set}}$ closed under limits. The claim for the right-hand variable comes down to the fact that polynomials are sums of representables; representable functors commute with all limits and sums commute with connected limits in **Set**. See [**kock2012polynomial**] for details. □

Exercise 2.40. Use Theorem 2.39 in the following.

1. Let p be a polynomial, thought of as a functor $p\colon \mathbf{Set} \to \mathbf{Set}$. Show that p preserves connected limits (of sets).
2. Show that for any polynomials p, q, r we have an isomorphism:
$$p \triangleleft (qr) \cong (p \triangleleft q) \times_{(p \triangleleft 1)} (p \triangleleft r)$$
3. Show that the distributivity $pq \triangleleft r \cong (p \triangleleft r)(q \triangleleft r)$ is a special case of Theorem 2.39.
4. Show that for any set A and polynomials p, q, we have an isomorphism $A(p \triangleleft q) \cong (Ap) \triangleleft q$. ◊

While we're here, it will be helpful to record the following.

Proposition 2.41. For any polynomial $q \in \mathbf{Poly}$, tensoring with q (on either side) preserves connected limits. That is, if J is connected and $p\colon J \to \mathbf{Poly}$ is a functor, then there is a natural isomorphism:

$$\left(\lim_{j\in J} p_j\right) \otimes q \cong \lim_{j\in J}(p_j \otimes q).$$

Proposition 2.42. For any polynomials p, p', q, q' there are natural maps

$$(p \triangleleft p') + (q \triangleleft q') \to (p+q) \triangleleft (p'+q') \tag{2.43}$$
$$(p \triangleleft p') \otimes (q \triangleleft q') \to (p \otimes pq) \triangleleft (p' \otimes q') \tag{2.44}$$
$$(p \triangleleft p') \times (q \triangleleft q') \leftarrow (p \times q) \triangleleft (p' \times q') \tag{2.45}$$

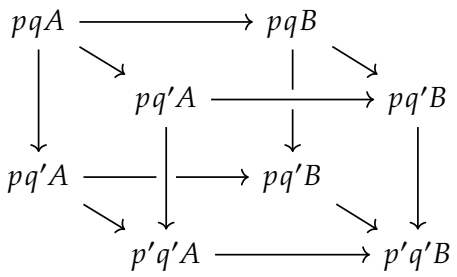
making $(+, \triangleleft)$ and $(\otimes, \triangleleft)$ duoidal structures and $(\times, \triangleleft)$ op-duoidal.

Proof. For (2.43) we have inclusion maps $p \to p+q$ and $p' \to p'+q'$, inducing a map $p \triangleleft p' \to (p+q) \triangleleft (p'+q')$. Similarly we obtain a map $q \triangleleft q' \to (p+q) \triangleleft (p'+q')$, so we get the desired map from the universal property of coproducts. It is straightforward to check that this is duoidal. The result for (2.45) is similar.

It remains to give a map (2.45).** □

Proposition 2.46 ($\triangleleft$ preserves cartesian maps in both variables)**.** If $f\colon p \to p'$ and $g\colon q \to q'$ are cartesian then so is $(f \triangleleft g)\colon (p \triangleleft q) \to (p' \triangleleft q')$.

Proof. For any $h\colon A \to B$ of sets, all faces of the cube



are pullbacks by Proposition 1.219 and Theorem 2.39. Hence the diagonal is too by standard properties of pullbacks. □

Exercise 2.47.

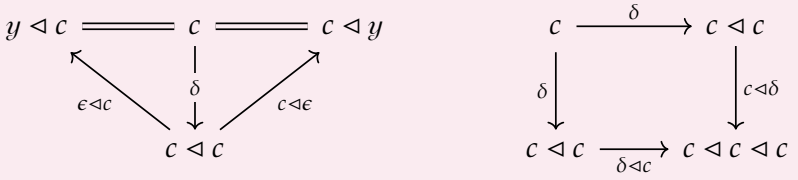
1. Show that if f is an isomorphism and g is vertical then $f \triangleleft g$ is vertical.
2. Find a polynomial q and a vertical morphism $f\colon p \to p'$ such that $(f \triangleleft \mathrm{id}_q)\colon (p \triangleleft q) \to (p' \triangleleft q)$ is not vertical. ◊

2.2 Comonoids in Poly

Imagine a sort of realm, where there are various positions you can be in. From every position, there are a number of moves you can make, possibly infinitely many. But whatever move you make, you'll end up in a new position. Well, technically it counts as a move to simply stay where you are, so you might end up in the same position. But wherever you move to, you can move again, and any number of moves from an original place counts as a single move. What sort of realm is this?

The most surprising aspects of **Poly** really begin with its comonoids. In 2018, researchers Daniel Ahman and Tarmo Uustalu showed that comonoids in $(\mathbf{Poly}, y, \triangleleft)$ can be identified with categories. Every category in the usual sense is a comomonoid in **Poly** and every comonoid in **Poly** is a category. We find this revelation to be truly shocking, and suggests some very different ways to think about categories. Let's go through it.

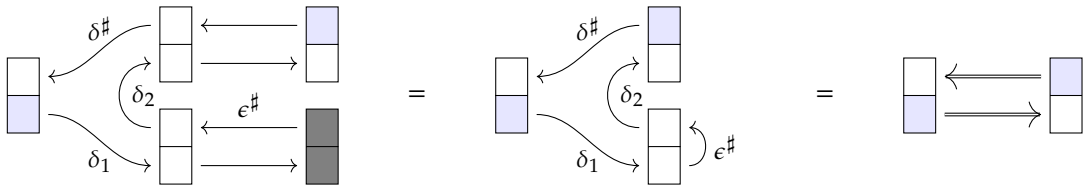
Definition 2.48 (Comonoid)**.** A *comonoid* in a monoidal category $(\mathcal{C}, I, \triangleleft)$ is a tuple (c, ϵ, δ) where $c \in \mathcal{C}$ is an object, and $\epsilon\colon c \to I$ and $\delta\colon c \to c \triangleleft c$ are maps, such that the following diagrams commute:



(2.49)

We refer to a comonoid $P := (p, \epsilon, \delta)$ in $(\mathbf{Poly}, y, \triangleleft)$ as a *polynomial comonoid*.

Here's a picture of one of the unit laws:



We'll put the associativity and the other unitality picture in the following exercise. The meaning of ϵ and δ will become clear; for those who want a hint, see **??**.

Exercise 2.50.

1. Draw the other unitality equation.
2. Draw the associativity equation. ◊

Example 2.51 (δ^n notation). Let (c, ϵ, δ) be a comonoid. From the associativity of δ, the two ways to get a map $c \to c \triangleleft c \triangleleft c$ have the same result. This is true for any $n \in \mathbb{N}$: we get an induced map $c \to c^{\triangleleft n+1}$, which by mild abuse of notation we denote δ^n:

$$c \xrightarrow{\delta} c \triangleleft c \xrightarrow{c \triangleleft \delta} c \triangleleft c \triangleleft c \xrightarrow{c^{\triangleleft 2} \triangleleft \delta} \cdots \xrightarrow{c^{\triangleleft n} \triangleleft \delta} c^{\triangleleft (n+1)}.$$

In particular, we have $\delta^1 = \delta$ and we may write $\delta^0 := \mathrm{id}_c$ and $\delta^{-1} := \epsilon$.

Polynomial comonoids are usually called *polynomial comonads*. Though polynomials p can be interpreted as polynomial *functors* $p : \mathbf{Set} \to \mathbf{Set}$, we do not generally emphasize this part of the story; we use it when it comes in handy, but generally we think of polynomials more as dependent arenas, or sets of corollas.

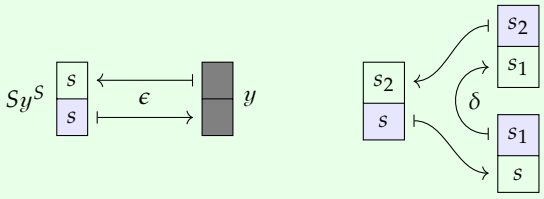
Example 2.52 (The state comonad Sy^S). Let S be a set, and consider the polynomial $p := Sy^S$. It has a canonical comonoid structure—often called the *state* comonad—as we discussed in Section 1.3, page 35. To say it in the current language, we first need to give maps $\epsilon : p \to y$ and $\delta : p \to p \triangleleft p$. By Eqs. (2.4) and (2.17), this is equivalent to giving functions

$$S \xrightarrow{\epsilon'} S \qquad\qquad S \xrightarrow{\delta'} \sum_{s' \in S} \prod_{s_1 \in S} \sum_{s'_1 \in S} \prod_{s_2 \in S} S$$

We take ϵ' to be the identity and we take δ' to be

$$s \xmapsto{\epsilon'} s \qquad\qquad s \xmapsto{\delta'} (s' := s, s_1 \mapsto (s'_1 := s_1, s_2 \mapsto s_2)). \tag{2.53}$$

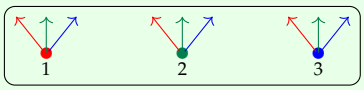
If you know how to read such things, you'll see that each element $s \in S$ is just being passed in a straightforward way. But we find this notation cumbersome and prefer the poly-box notation.



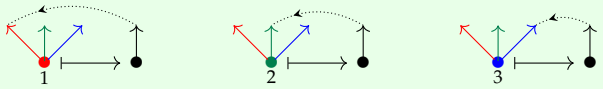
(2.54)

Exercise 2.55. Let $p := Sy^S$. For any $n \in \mathbb{N}$, write out the morphism of polynomials $\delta^n : p \to p^{\triangleleft (n+1)}$ either set-theoretically or in terms of poly-boxes as in Example 2.52 ◊

Example 2.56 (Picturing the comonoid Sy^S). Let's see this whole thing in pictures. First of all, let's take $S := 3 \cong \{\bullet, \bullet, \bullet\}$ and draw $\{\bullet, \bullet, \bullet\}y^{\{\bullet,\bullet,\bullet\}}$:

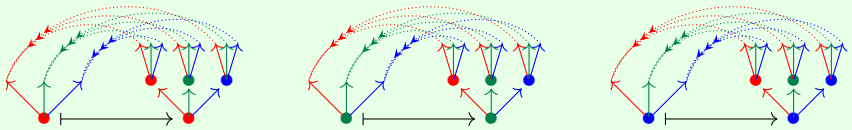
$3y^3 =$ 

The map $\epsilon : Sy^S \to y$ can be drawn as follows:



It picks out one direction at each position, namely the one of the same color.

The map $\delta : Sy^S \to (Sy^S)^{\triangleleft 2}$ can be drawn as follows:



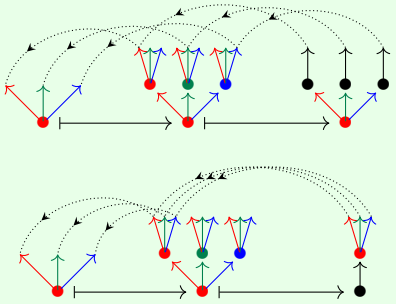
(2.57)

Note that $(Sy^S)^{\triangleleft 2}$ has SS^S, or in this case 81 many trees, only three of which are being pointed to by δ. That is, there is in general no rule on trees in that says the color of an arrow should agree in any sense with the color of the node it points to: (2.57) shows that the comonoid structure is pointing out the special trees where that does occur.

It remains to check the comonoid laws, the three commutative diagrams in (2.49). The first two say that the composites

$$Sy^S \xrightarrow{\delta} (Sy^S)^{\triangleleft 2} \xrightarrow{\mathrm{id} \triangleleft \epsilon} Sy^S \qquad \text{and} \qquad Sy^S \xrightarrow{\delta} (Sy^S)^{\triangleleft 2} \xrightarrow{\epsilon \triangleleft \mathrm{id}} Sy^S$$

are the identity. Let's return to the case $S = 3$. Then the second map in each case involves 81 different assignments, but only three of them will matter.[a] Since all three are strongly similar, we will draw only the red case. We also only draw the relevant passback maps.



We do not show associativity here, but instead leave it to the reader in Exercise 2.58.

[a] To say technically that we can disregard all but three positions in $(Sy^S)^{\triangleleft 2} \cong SS^S y^{SS}$, one can use Proposition 1.216.

Exercise 2.58. Let $S := 2$ and $c := 2y^2$.

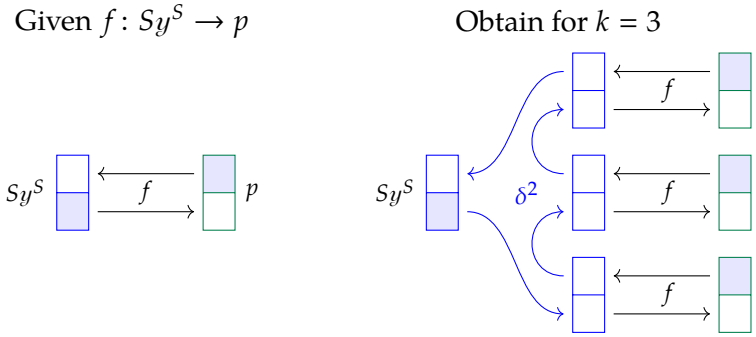
1. Draw c using color if possible.
2. We know c is supposed to be the carrier of a comonoid (c, ϵ, δ). Which two maps $c \to c^{\triangleleft 3}$ are supposed to be equal by associativity?
3. Draw these two maps in the style of Example 2.56.
4. Are they equal? ◊

Speeding up dynamical systems Suppose we have a dynamical system $f : Sy^S \to p$, and we want to make it go k-times faster. That is, in every moment, we want it to process k-many inputs, rather than one.

Since Sy^S has the structure of a comonoid, we know that for every $k \in \mathbb{N}$ we have a map $\delta^{k-1} : Sy^S \to (Sy^S)^{\triangleleft k}$ by Example 2.51. But we also have maps $f^{\triangleleft k} : (Sy^S)^{\triangleleft k} \to p^{\triangleleft k}$ because $\triangleleft$ is a monoidal product. Thus we can form the composite

$$Sy^S \xrightarrow{\delta^{k-1}} (Sy^S)^{\triangleleft k} \xrightarrow{f^{\triangleleft k}} p^{\triangleleft k} \qquad \text{Spdup}_k(f) \tag{2.59}$$

For every state $s \in S$, we now have a length-k strategy in p, i.e. a tree of height k in p, or explicitly a choice of p-position and, for every direction there, another p-position and so on k times. Here is a poly-box drawing for the $k = 3$ case:



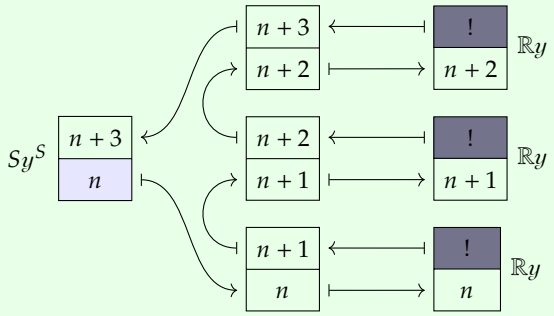
Example 2.60. Let $p := \mathbb{R}y^1$, let $S := \mathbb{N}$, and let $\begin{pmatrix} f^\sharp \\ f_1 \end{pmatrix} : Sy^S \to p$ be given by $f_1(n) := n$ and $f^\sharp(n, 1) := n + 1$. What is this speedup map $\text{Spdup}_k(f)$? First of all, its type is

$$\text{Spdup}_k(f) : Sy^S \to \mathbb{R}^k y,$$

meaning that it has the same set of states as before, but it outputs k-many reals in every

moment.

So for example with $k = 3$ here is one moment of output:



So for example starting at initial state $n = 0$, we get the following output stream, e.g. for 4 seconds:

$$(0,1,2),(3,4,5),(6,7,8),(9,10,11).$$

Other comonoids Once you know that these all important Sy^S-things are comonoids in **Poly**, it's interesting to ask "what are all the comonoids in **Poly**?" Let's discuss another one before answering the question in generality.

Example 2.61 (A simple comonoid that's not Sy^S). The polynomial $y^2 + y$ can be given a comonoid structure. Let's first associate names to its positions and directions.

Define $w \coloneqq \{A\}y^{\{i_A, f\}} + \{B\}y^{\{i_B\}}$; it is clearly isomorphic to $y^2 + y$, but its notation is meant to remind the reader of the walking arrow category

$$\mathcal{W} \coloneqq \boxed{A \xrightarrow{f} B}$$

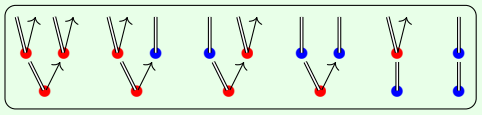
We will use the category $\mathcal{W}$ as inspiration for equipping w with a comonoid structure (w, ϵ, δ). The map ϵ will pick out identity arrows and the map δ will tell us about codomains and composition (which is rather trivial in the case of $\mathcal{W}$). Here's a picture of $w \cong y^2 + y$:

$$w \coloneqq \boxed{\begin{array}{ccc} i_A & & i_B \\ & f & \\ A & & B \end{array}}$$

We first need to choose a map of polynomials $\epsilon \colon w \to y$; it can be identified with a dependent function $\epsilon^\sharp \colon (o \in w(1)) \to w[o]$, assigning to each position a direction there. Let's take $\epsilon^\sharp(A) \coloneqq i_A$ and $\epsilon^\sharp(B) \coloneqq i_B$:

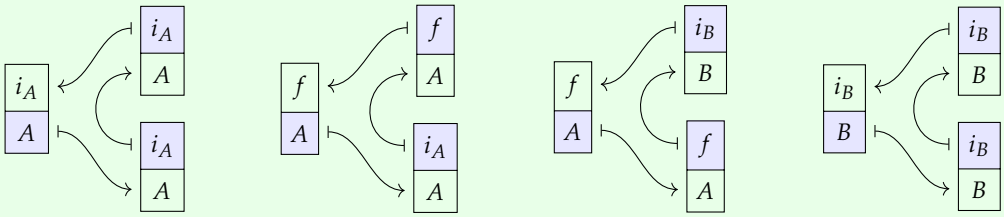
Now we need a map of polynomials $\delta \colon w \to w \triangleleft w$. Let's draw out $w \triangleleft w$ to see what it

looks like.

$w \triangleleft w =$ 

The map δ is going to tell us both about codomains and composition. Here it is:

The on-positions map selects, for each position (either A or B) the two-level tree starting at that position and having the correct codomains: the identity arrow on A points to the corolla for A; the f map points to the corolla for B; and the identity arrow on B points to the corolla for B. The on-directions maps assign the correct composites. Here is $\delta\colon w \to w \triangleleft w$ again, in terms of poly-boxes.



It remains to check that (w, ϵ, δ) really is a comonoid, i.e. that the diagrams in (2.49) commute. We will check unitality only for A; it is easier for B.

In both pictures, one can see that the composite map is the identity. We would do associativity here, but because the category $\mathcal{W}$ is so simple, associativity is guaranteed; this makes the pictures too trivial.

Exercise 2.62. Write out the data (c, ϵ, δ) for the comonoid corresponding to the category

$$B \xleftarrow{f} A \xrightarrow{g} C$$

For this exercise, you are not being asked to check the unitality or associativity conditions. ◊

Exercise 2.63. Show that if A is a set and $p := Ay$ is the associated linear polynomial, then there exists a unique comonoid structure on p. ◊

Example 2.64 (The category of A-streams). For any set A, the set $A^{\mathbb{N}}$ of A-streams

$$s = (a_0 \rightsquigarrow a_1 \rightsquigarrow a_2 \rightsquigarrow a_3 \rightsquigarrow \cdots)$$

are the objects of a category, where the set of morphisms emanating from each stream $s \in A^{\mathbb{N}}$ is $\mathbb{N}$. The identity on s is given by 0 and the composite of two morphisms is the sum of the corresponding natural numbers.

We will see this category again in Example 2.147.

2.2.1 Comonoids in Poly are categories

It turns out that comonoids in **Poly** are precisely categories. Strangely, however, the a morphism between comonoids is not a functor but something people are calling a *cofunctor*.

Definition 2.65 (Cofunctor)**.** Let $\mathcal{C}$ be a category with object set C_0, morphism set C_1, $\mathrm{dom}, \mathrm{cod} : C_1 \to C_0$ the domain and codomain,[a] and similarly for $\mathcal{D}$. A *cofunctor* $F : \mathcal{C} \nrightarrow \mathcal{D}$ consists of

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

satisfying the following conditions:

i. $F^\sharp(c, \mathrm{id}_{F_0 c}) = \mathrm{id}_c$ for any $c \in C_0$;
ii. $F_0 \,\mathrm{cod}\, F^\sharp(c, g) = \mathrm{cod}\, g$ for any $c \in C_0$ and $g \in D_{F_0(c)}$;
iii. $F^\sharp(\mathrm{cod}\, F^\sharp(c, g_1), g_2) \circ F^\sharp(c, g_1) = F^\sharp(c, g_1 \fatsemi g_2)$ for composable arrows g_1, g_2 out of $F_0 c$.

In other words, $F^\sharp$ preserves identities, codomains, and compositions.

We denote by $\mathbf{Cat}^\sharp$ the category of categories and cofunctors.

[a] We privilege the domain function $\mathrm{dom} : C_1 \to C_0$ in the sense that an unnamed map $C_1 \to C_0$ will be assumed to be dom.

The cofunctor laws can be written in commutative diagram form as follows:

$$\begin{array}{ccc} C_0 \times_{D_0} D_0 & \xrightarrow{\cong} & C_0 \\ {\scriptstyle \mathrm{id}_D}\downarrow & \text{(i)} & \downarrow{\scriptstyle \mathrm{id}_C} \\ C_0 \times_{D_0} D_1 & \xrightarrow[F^\sharp]{} & C_1 \end{array} \qquad \begin{array}{ccccc} C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} & C_1 & \xrightarrow{\mathrm{cod}} & C_0 \\ {\scriptstyle \pi_2}\downarrow & \text{(ii)} & & & \downarrow{\scriptstyle F_0} \\ D_1 & & \xrightarrow[\mathrm{cod}]{} & & D_0 \end{array}$$

$$\begin{array}{ccccc} C_0 \times_{D_0} D_1 \times_{D_0} D_1 & \xrightarrow{\fatsemi_D} & C_0 \times_{D_0} D_1 & \xrightarrow{F^\sharp} & C_1 \\ {\scriptstyle F^\sharp}\downarrow & & \text{(iii)} & & \uparrow{\scriptstyle \fatsemi_C} \\ C_1 \times_{D_0} D_1 & \xrightarrow[\cong]{} & C_1 \times_{C_0} C_0 \times_{D_0} D_1 & \xrightarrow[F^\sharp]{} & C_1 \times_{C_0} C_1 \end{array}$$

Example 2.66 (Systems of ODEs). A system of ordinary differentrial equations (ODEs) in n variables, e.g.

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, \ldots, x_n) \\ \dot{x}_2 &= f_2(x_1, \ldots, x_n) \\ &\vdots \\ \dot{x}_n &= f_n(x_1, \ldots, x_n) \end{aligned}$$

can be understood as a vector field on $\mathbb{R}^n$. Usually, we are interested in integrating this vector field to get flow lines, or integral curves. In other words, for each point $x = (x_1, \ldots, x_n)$ and each amount of time $t \in \mathbb{R}$, we can go forward from x for time t and arrive at a new point x^{+t}. These satisfy the equations

$$x^{+0} = x \qquad \text{and} \qquad x^{+t_1+t_2} = (x^{+t_1})^{+t_2}. \tag{2.67}$$

Let's call such things *dynamical systems* with time domain $(T, 0, +)$; above, we used $T = \mathbb{R}$, but any monoid will do.

Dynamical systems in the above sense are cofunctors $F\colon \mathbb{R}^n y^{\mathbb{R}^n} \nrightarrow y^T$. In order to say this, we first need to say how both $\mathcal{C} := \mathbb{R}^n y^{\mathbb{R}^n}$ and y^T are being considered as categories. The category $\mathcal{C}$ has objects $\mathbb{R}^n$, and for each object $x \in \mathbb{R}^n$ and outgoing arrow $v \in \mathbb{R}^n$, the codomain of v is $x + v$; in other words, v is a vector emanating from x. The identity is $v = 0$, and composition is given by addition. The category y^T is the monoid T considered as a category with one object, $\bullet$.

The cofunctor assigns to every object $x \in \mathbb{R}^n$ the unique object $F(x) = \bullet$, and to each element $t \in T$ the morphism $F^\sharp(x, t) = x^{+t} - x \in \mathbb{R}^n$, which can be interpreted as a vector emanating from x. Its codomain is $\operatorname{cod} F^\sharp(x, t) = x^{+t}$, and we will see that (2.67) ensures the cofunctoriality properties.

The codomain law ii is vacuously true, since y^T only has one object. Law i follows because $F^\sharp(x, 0) = x^{+0} - x = 0$, and law iii follows as

$$F^\sharp(x^{+t_1}, t_2) + F^\sharp(x, t_1) = (x^{+t_1})^{+t_2} - x^{+t_1} + x^{+t_1} - x = x^{+t_1+t_2} - x = F^\sharp(x, t_1 + t_2).$$

Theorem 2.68 (Ahman-Uustalu)**.** There is an equivalence of categories

$$\mathbf{Comon}(\mathbf{Poly}) \cong \mathbf{Cat}^\sharp.$$

Proof. This will be proved as Proposition 2.153 and Theorem 2.159. □

Our first goal is to understand how one translates between categories $\mathcal{C}$ and comonoids $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ in **Poly**. The idea is pretty simple: the objects of $\mathcal{C}$ are the positions of $\mathfrak{c}$

$$\mathrm{Ob}(\mathcal{C}) \cong \mathfrak{c}(1)$$

and for each such object i, the morphisms $\{f : i \to j \mid j \in \mathrm{Ob}(\mathcal{C})\}$ emanating from i in $\mathcal{C}$ are the directions $\mathfrak{c}[i]$ there.

Definition 2.69. Let $\mathcal{C}$ be a category. The *emanation polynomial for* $\mathcal{C}$ is the polynomial

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

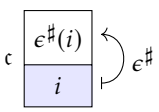
Exercise 2.70. What is the emanation polynomial for each of the following categories?

1. $\boxed{A \xrightarrow{f} B}$?
2. $\boxed{B \xleftarrow{f} A \xrightarrow{g} C}$?
3. The empty category?
4. The monoid $(\mathbb{N}, 0, +)$?
5. A monoid $(M, e, *)$?
6. The poset $(\mathbb{N}, \leq)$?
7. The poset $(\mathbb{N}, \geq)$? ◊

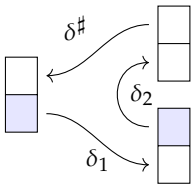
A category $\mathcal{C}$ is more than its emanation polynomial $\mathfrak{c}$, and a comonoid $(\mathfrak{c}, \epsilon, \delta)$ in **Poly** is more than its carrier polynomial $\mathfrak{c}$. The identities of $\mathcal{C}$ are all captured by the counit $\epsilon : \mathfrak{c} \to y$ and the codomain and composition information of $\mathcal{C}$ are all captured by the comultiplication map $\delta : \mathfrak{c} \to \mathfrak{c} \triangleleft \mathfrak{c}$. Our goal is to make this clear so that we can justly proclaim:

Comonoids in ***Poly*** *are precisely categories!*

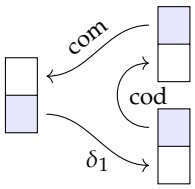
We want to understand how the counit ϵ and comultiplication δ in a comonoid $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ relate to identities, codomains, and composites in a category. We first use our work in Section 2.1.3 to get a better handle on ϵ and δ. For example, since $\epsilon : \mathfrak{c} \to y$ maps to the empty composite, we know by (2.26) that it is of the form



i.e. for every $i \in \mathfrak{c}(1)$, a choice of element $\epsilon^\sharp(i) \in \mathfrak{c}[i]$. Rather than call it $\epsilon^\sharp$, we will refer to this map as idy. Similarly, we know by (2.27) that δ is of the form

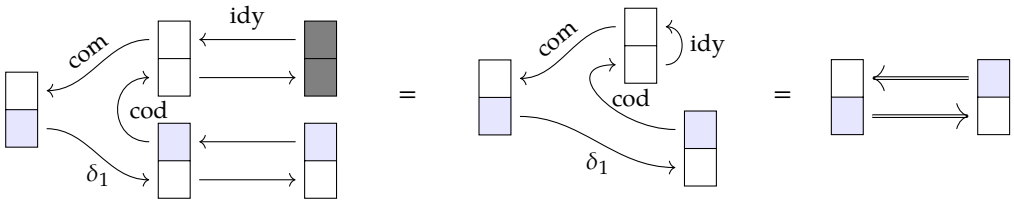


We've said that this secretly holds information about the codomains and composites for a category structure on $\mathfrak{c}$. How does that work? We will soon find that δ_1 is forced to be an identity, that δ_2 holds codomain information, and that $\delta^\sharp$ holds composite information. So we will use that notation here

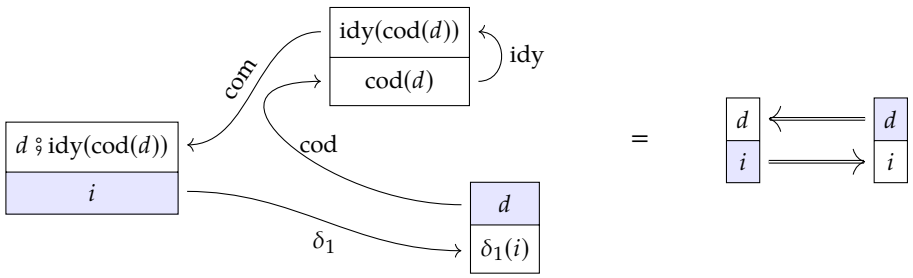


and our goal now is to see that the cod map really has something to do with codomains and that the com map really has something to do with composites, as advertised. What makes these true are the unitality and associativity equations required for $(\mathfrak{c}, \epsilon, \delta)$ to be a comonoid; see Definition 2.48.

To get started, we consider the first unitality equation from (2.49):



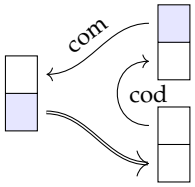
Let's add some arbitrary fillers $i \in \mathfrak{c}(1)$ and $d \in \mathfrak{c}[i]$ to the open slots, and hence obtain an equation:



First it's saying that $\delta_1(i) = i$. This is great news; it means we can forget about δ_1, just as we said earlier. Second it's saying that $d \fatsemi \text{idy}(\text{cod}(d)) = d$. Unpacking, this means

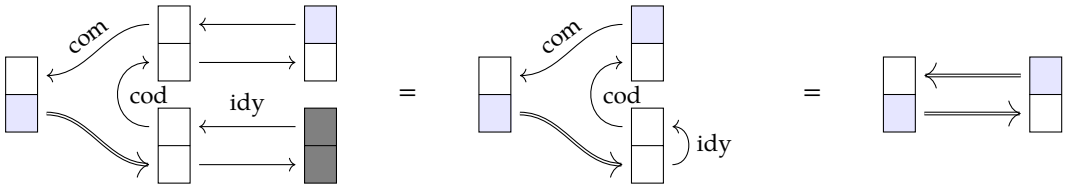
that composing a morphism d with the identity morphism on its codomain returns d. It's neat to watch the comonoid laws declaring the standard laws of categories. It's like meeting a like-minded toad; we never knew toads could be like-minded, but the phenomena don't lie.

Before moving on, we redraw $\delta : c \to c \triangleleft c$ with the information-lacking δ_1 (which the first unitality equation said was always identity) and replace it with a double arrow:

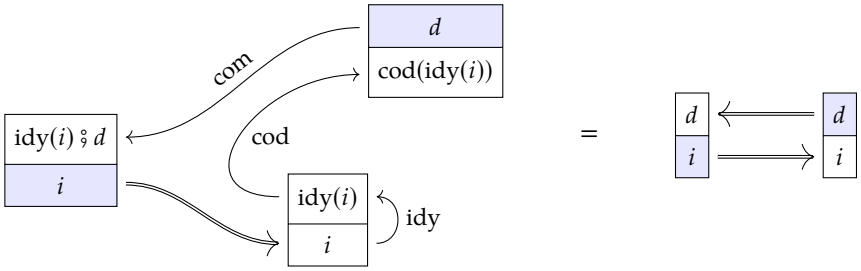


(2.71)

Now we can write the other unitality equation.

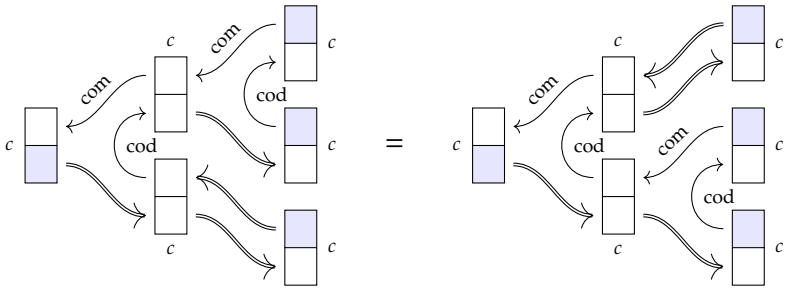


Let's add some arbitrary fillers $i \in c(1)$ and $d \in c[i]$ to get some equations:

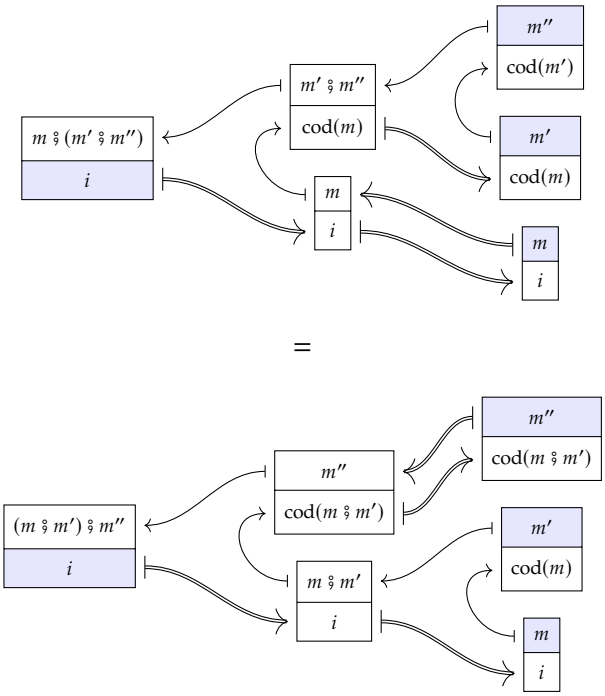


Ah, it's saying that it wants $\mathrm{cod}(\mathrm{idy}(i)) = i$, which makes sense—the codomain of the identity on i should be i—and that it wants $\mathrm{idy}(i) \fatsemi d = d$, i.e. that composing the identity on i with d should return d. We couldn't have said it better ourselves; thanks like-minded toad!

Finally we draw the associativity equation.



Let's fill it in with $i \in c(1)$ and a sequence $i \xrightarrow{m} \xrightarrow{m'} \xrightarrow{m''}$ of emanating morphisms:



Ah, it's saying that it wants $\mathrm{cod}(m') = \mathrm{cod}(m \fatsemi m')$; well yeah, that's how codomains should work. And it wants $m \fatsemi (m' \fatsemi m'') = (m \fatsemi m') \fatsemi m''$, classic associativity. Amazing; thanks again toad!

We've seen that all of the data and equations of categories are embedded, though in a very non-standard way, in the data and equations of polynomial comonoids.

Exercise 2.72. Let $\mathcal{C}$ be a category, $\mathfrak{c}$ its emanation polynomial, and $i \in \mathrm{Ob}(\mathcal{C})$ an object. This exercise is for people who know the definition of the coslice category $i/\mathcal{C}$ of objects under i. Is it true that there is an isomorphism

$$\mathrm{Ob}(c/\mathcal{C})i \cong^{?} \mathfrak{c}[i]$$

If so, describe it; if not, give a counterexample. ◊

2.2.2 Examples showing the correspondence between comonoids and categories

Example 2.73 (Monoids). Let $(M, e, *)$ be a monoid. Then we can construct a comonoid structure on the representable y^M. A morphism $y^M \to y$ can be identified with an element of M; under that identification we take $\epsilon := e$. Similarly, $y^M \triangleleft y^M \cong y^{M^2}$

and a morphism $y^M \to y^{M^2}$ can be identified with a function $M^2 \to M$; under that identification we take $\delta := *$.

Exercise 2.74. Finish Example 2.73 by showing that if $(M, e, *)$ satisfies the unitality and associativity requirements of a monoid in $(\mathbf{Set}, 1, \times)$ then (y^M, ϵ, δ) satisfies the unitality and associativity requirements of a comonoid in $(\mathbf{Poly}, y, \triangleleft)$. ◊

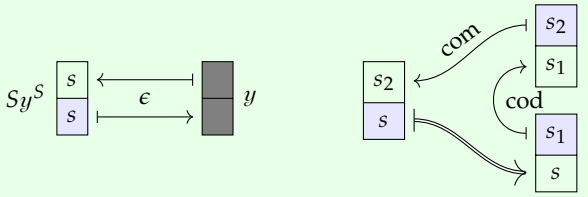
Example 2.75 (Cyclic lists). For any $n \in \mathbb{N}$, consider the monoid (group) $\mathbb{Z}/n$. As a functor $c_n := y^{\mathbb{Z}/n}$ sends a set X to the set of length-n tuples in X. But the comonoid structure lets us think of these as cyclic lists. Indeed, $\epsilon : c_n \to y$ allows us to pick out the "current" element via the map $\epsilon \triangleleft X : c_n \triangleleft X \to X$, and δ lets us move around the list.

We will see later that comonoids are closed under coproducts, so $\sum_{n \in \mathbb{N}} c_n$ is also a comonoid.

Example 2.76 (What category is Sy^S?). The first comonoid we introduced, back in Example 2.52 was $\mathscr{S} = (p, \epsilon, \delta)$, where $p = Sy^S$ for some set S. Now we know that comonoids correspond to categories. So what category $\mathcal{S}$ corresponds to $\mathscr{S}$?

By the work above, we know that $\mathcal{S}$ has object set $S = p(1)$, and that for every object $s \in S$ there are S-many emanating morphisms, though we don't yet know their codomains nor the composition formula.

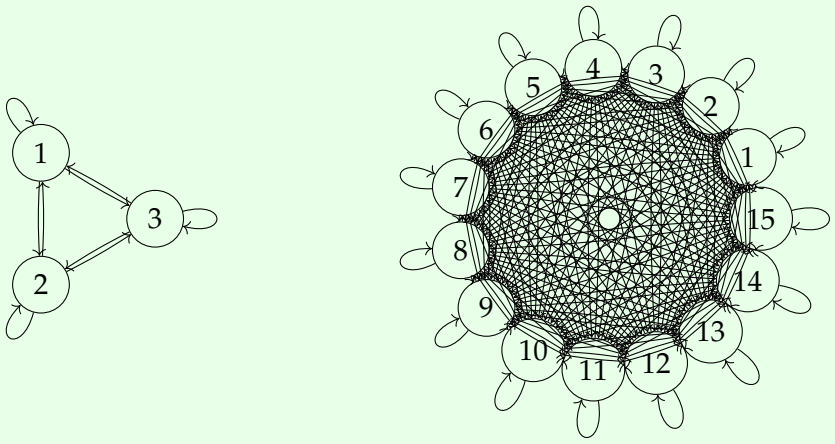
To calculate the codomains and compositions we examine the map $\delta : p \to p \triangleleft p$, which was given set-theoretically in (2.53) and in terms of poly-boxes in (2.54). We repeat it here for your convenience:



The ϵ map is saying that the identity on the object s is the emanating morphism s. Remember that both the set of objects and the set of morphisms emanating from any given object are S. The map cod $= \delta_2$ is telling us that the codomain of the morphism s_1 emanating from s is the object s_1, and that the composite of s_1 and s_2 is $s_1 \fatsemi s_2 = s_2$.

What this all means is that $\mathcal{S}$ is the category with S-many objects and a unique morphism $s \to s'$ for any $s, s' \in S$. Here are pictures for $S = 3$ and $S = 15$, with all

maps (even identities) drawn:



Some people would call this the contractible groupoid, or the terminal category with S-elements, or the unique category whose underlying graph is complete on S vertices. The one that's *least* good for us will be "terminal" category, because as we'll see, we're going to be interested in different sorts of morphisms between categories than the usual ones, namely cofunctors rather than functors, and $\mathcal{S}$ is not terminal for cofunctors.

Anyway, to avoid confusion, we'll refer to $\mathcal{S}$ as the *state category on* S, because we will use these to think about states of dynamical systems, and also because the state comonad in functional programming is Sy^S.

Exercise 2.77. We showed in Exercise 2.63 that for any set A, the linear polynomial $p \coloneqq Ay$ has a unique comonoid structure. What category does it correspond to? ◊

Definition 2.78. Let $\mathcal{C}$ be a category and $c \in \mathrm{Ob}(\mathcal{C})$ an object. The *degree of* c, denoted $\deg(c)$ is the set of arrows in $\mathcal{C}$ that emanate from c.

If $\deg(c) \cong 1$, we say that c is *linear* and if $\deg(c) \cong \mathsf{n}$ for $n \in \mathbb{N}$, we say c has *degree* n.

Exercise 2.79.

1. If every object in $\mathcal{C}$ is linear, what does it mean about $\mathcal{C}$?
2. Is it possible for an object in $\mathcal{C}$ to have degree 0?
3. Find a category that has an object of degree $\mathbb{N}$.
4. How many categories are there that have just one linear and one quadradic (degree 2) object?
5. Is the above the same as asking how many comonad structures on $y^2 + y$ there

are?

◊

Exercise 2.80.

1. Find a category structure for the polynomial $y^{n+1} + ny$.
2. Would you call your category "star-shaped"? ◊

Exercise 2.81. Let S be a set. Is there any comonoid structure on Sy^S other than that of the state category? ◊

2.2.3 Morphisms of comonoids are cofunctors

Our next goal is to understand morphisms $\mathscr{C} \to \mathscr{D}$ between comonoids and what they look like as maps between categories $\mathcal{C} \to \mathcal{D}$.

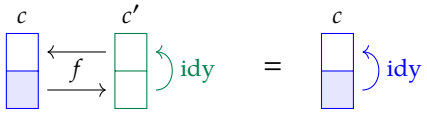
Cofunctors F are forward on objects, and backwards on morphisms. It's good to remember: Codomains are objects, so F preserves them going forwards; identities and composites are morphisms, so F preserves them going backwards.

Let's begin with a definition.

Definition 2.82 (Morphisms of comonoids)**.** Let $\mathscr{C} := (c, \epsilon, \delta)$ and $\mathscr{C}' := (c', \epsilon', \delta')$ be polynomial comonoids as in Definition 2.48. A *morphism* $\mathscr{C} \to \mathscr{C}'$ consists of a morphism $f : c \to c'$ of polynomials that commutes with the structure maps:

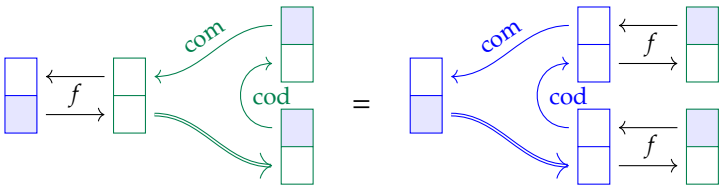
$$\begin{array}{ccc} c & \xrightarrow{f} & c' \\ {\scriptstyle\epsilon}\downarrow & & \downarrow{\scriptstyle\epsilon'} \\ y & = & y \end{array} \qquad \begin{array}{ccc} c & \xrightarrow{f} & c' \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\delta'} \\ c \triangleleft c & \xrightarrow[f \triangleleft f]{} & c' \triangleleft c' \end{array} \tag{2.83}$$

Let's see the two laws of comonoid morphisms using poly-boxes. First the counit law:



(2.84)

Then the comultiplication law:



(2.85)

If we fill in Eq. (2.84) with an object $i \in \mathfrak{c}(1)$, we obtain the equation

$$f^\sharp(i, \mathrm{idy}(f_1(i))) = \mathrm{idy}(i),$$

which is the first law of Definition 2.65. If we fill in Eq. (2.85) with $i \in \mathfrak{c}(1)$ and $m \in \mathfrak{c}'[f(i)]$ and $m' \in \mathfrak{c}'[\mathrm{cod}(m)]$, we obtain the equations

$$\mathrm{cod}(m) = f_1\big(\mathrm{cod}\,\big(f^\sharp(i, m)\big)\big)$$
$$f^\sharp(i, \mathrm{com}(m, m')) = \mathrm{com}(f^\sharp(i, m), f^\sharp(\mathrm{cod}(f^\sharp(i, m)), m'))$$

and these are the second and third laws of Definition 2.65.

Exercise 2.86. Summarize the proof of Theorem 2.68, developed above. You may cite anything written in the text so far. ◊

Proposition 2.87. Let $F\colon \mathcal{C} \nrightarrow \mathcal{D}$ be a cofunctor, $c, c' \in \mathrm{Ob}(\mathcal{C})$ objects, and $g\colon F(c) \to F(c')$ a morphism in $\mathcal{D}$. Then if g is an isomorphism, so is $F_c^\sharp(g)$.

Proof. With $d \coloneqq F(c)$, $d' \coloneqq F(c')$, and g' the inverse of g, we have

$$\begin{aligned} \mathrm{id}_c &= F_c^\sharp(\mathrm{id}_d) \\ &= F_c^\sharp(g \fatsemi g') \\ &= F_c^\sharp(g) \fatsemi F_{c'}^\sharp(g') \end{aligned}$$

Thus $F_c^\sharp(g)$ is a section of $F_{c'}^\sharp(g')$. The opposite is true similarly, completing the proof. □

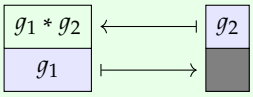
Examples of cofunctors. We saw in Theorem 2.68, summarized in Exercise 2.86, that cofunctors $\mathcal{C} \nrightarrow \mathcal{D}$ are the same thing as morphisms of comonoids $\mathscr{C} \to \mathscr{D}$, so we elide the difference. The question we're interested in now is: how do we think about cofunctors? What is a map of polynomial comonoids like?

The rough idea is that a cofunctor $\mathcal{C} \nrightarrow \mathcal{D}$ is, in particular, a morphism $\mathfrak{c} \to \mathfrak{d}$ in **Poly** between their emanation polynomials. This map preserves identities, codomains, and composition, which is great, but you still feel like you've got a map of polynomials on your hands: it goes forwards on objects and backwards on morphisms.

If a functor $\mathcal{C} \to \mathcal{D}$ is a picture *of $\mathcal{C}$ in $\mathcal{D}$, then a cofunctor $\mathcal{C} \nrightarrow \mathcal{D}$ is a D-*shaped crystallization *of C.*

Let's look at some examples to see how cofunctors look like crystallizations, or perhaps partitions.

Example 2.88. Let $(G, e, *)$ be a group and (y^G, ϵ, δ) the corresponding comonoid. There is a cofunctor $Gy^G \to y^G$ given by



To see this is a cofunctor, we check that identities, codomains, and compositions are preserved. For any g_1, the identity e is passed back to $g_1 * e = g_1$, and this is the identity on g_1 in Gy^G. Codomains are preserved because there is only one object in y^G. Composites are preserved because for any g_2, g_3, we have $g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$.

Exercise 2.89. Does the idea of Example 2.88 work when G is merely a monoid, or does something go subtly wrong somehow? ◊

Proposition 2.90. There is a fully faithful functor $\mathbf{Mon}^{\mathrm{op}} \to \mathbf{Cat}^\sharp$, whose image is precisely those categories whose emanation polynomial is representable.

Proof. Given a monoid $(M, e, *)$, we think of it as a category with one object; its emanation polynomial y^M is representable. A cofunctor between such categories carries no data in its on-objects part, and codomains are automatically preserved. Cofunctors $y^M \to y^N$ simply carry elements of N to elements of M, preserving identity and composition, exactly the description of monoid homomorphisms. □

Proposition 2.91. There is an adjunction

$$\mathbf{Cat}^\sharp(\mathcal{C}, Ay) \cong \mathbf{Set}(\mathrm{Ob}(\mathcal{C}), A)$$

where $\mathcal{C} \in \mathbf{Cat}^\sharp$ is a comonoid and $A \in \mathbf{Set}$ is a set.

Example 2.92. Consider the category $\mathbb{R}y^{\mathbb{R}}$, where the codomain of r emanating from x is $x + r$, identities are 0, and composition is given by addition. What are cofunctors into $\mathbb{R}y^{\mathbb{R}}$?

Let $\mathcal{C}$ be a category and $|\cdot| \colon \mathcal{C} \nrightarrow \mathbb{R}y^{\mathbb{R}}$ a cofunctor. It assigns to every object c both a real number $|c| \in \mathbb{R}$ and a choice of emanating morphism $|c|^\sharp(r) \colon c \to c_r$ such that $|c| + r = |c_r|$. This assignment satisfies some laws. Namely we have $c_0 = c$ and, given reals $r, s \in \mathbb{R}$, we have $(c_r)_s = c_{r+s}$.

Exercise 2.93.

1. Do you think a cofunctor $\mathcal{C} \nrightarrow \mathbb{R}y^{\mathbb{R}}$ as in Example 2.92 should be called an $(\mathbb{R}, 0, +)$-action on the objects of $\mathcal{C}$, or a filtration, or a valuation, or something else?
2. Why? ◊

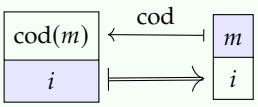
Exercise 2.94.

1. Over two discrete objects $\{A, B\}$, how many cofunctors are there

$$y^2 + y \cong \boxed{A \to B} \longrightarrow \boxed{A \rightrightarrows B} \cong y^3 + y$$

from the walking arrow category to the walking parallel-arrows category?
2. What is meant more precisely by "over two discrete objects $\{A, B\}$" above? ◊

Exercise 2.95. Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid in **Poly**. We have a state category $\mathfrak{c}(1)y^{\mathfrak{c}(1)}$ on the set of objects of $\mathscr{C}$. There is a map of polynomials $\mathfrak{c}(1)y^{\mathfrak{c}(1)} \to \mathfrak{c}$ given by



for an object $i \in \mathfrak{c}(1)$ and an outgoing morphism $m \in \mathfrak{c}[i]$. Is this map a cofunctor? ◊

Example 2.96 (Canonical cofunctors from state categories). Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid, where $\delta = (\mathrm{id}, \mathrm{cod}, \mathrm{com})$ as in (2.71). For any position $i \in \mathfrak{c}(1)$, there is a cofunctor

$$(\mathrm{cod}, \mathrm{com}) \colon \mathfrak{c}[i]y^{\mathfrak{c}[i]} \nrightarrow \mathfrak{c}.$$

That is, an object $f \in \mathfrak{c}[i]$ is also a morphism in $\mathcal{C}$ and we send it to its codomain $\mathrm{cod}(f)$. A morphism in $\mathcal{C}$ emanating from $\mathrm{cod}(f)$ is passed back to its composite with f.

Exercise 2.97. Suppose $\mathcal{C} = (\mathfrak{c}, \epsilon, \delta)$ is a comonoid.

1. Show that the map $(\mathrm{cod}, \mathrm{com}) \colon \mathfrak{c}[i]y^{\mathfrak{c}[i]} \nrightarrow \mathfrak{c}$ from Example 2.96 satisfies the conditions necessary for being a cofunctor (identities, codomains, and composites).
2. Find a comonoid structure on the polynomial $p := \sum_{i \in \mathfrak{c}(1)} \mathfrak{c}[i]y^{\mathfrak{c}[i]}$ and a cofunctor $p \to \mathfrak{c}$.
3. Is the polynomial map $p \to \mathfrak{c}$ an epimorphism? ◊

Exercise 2.98. Suppose $\mathfrak{c}, \mathfrak{d}, \mathfrak{e}$ are polynomials, each with a comonoid structure, and that $f : \mathfrak{c} \to \mathfrak{d}$ and $g : \mathfrak{d} \to \mathfrak{e}$ are maps of polynomials.

1. If f and $f \fatsemi g$ are each cofunctors, is g automatically a cofunctor? If so, sketch a proof; if not sketch a counterexample.
2. If g and $f \fatsemi g$ are each cofunctors, is f automatically a cofunctor? If so, sketch a proof; if not sketch a counterexample. ◊

Exercise 2.99.

1. For any category $\mathcal{C}$ with emanation polynomial $\mathfrak{c}$, find a category with emanation polynomial $\mathfrak{c}y$.
2. Show your construction is functorial; i.e. given a cofunctor $\mathfrak{c} \nrightarrow \mathfrak{d}$, find one $\mathfrak{c}y \nrightarrow \mathfrak{d}y$, preserving identity and composition.
3. Is your functor either a monad or a comonad on $\mathbf{Cat}^\sharp$?
4. What category do you get by repeatedly applying this functor to y? ◊

Exercise 2.100. Are cofunctors between posets interesting?

1. Consider the chain poset $[n] \cong \sum_{i=1}^{n} y^i$. How many cofunctors are there from $[m] \to [n]$ for all $m, n \in \{0, 1, 2, 3\}$?
2. What does a cofunctor from y into a poset represent? Is there anything you'd call "asymmetric" about it? ◊

Exercise 2.101.

1. What is the finite set $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\}$ of comonoids (defined up to isomorphism) for which the carrier polynomial is $y^2 + y$?
2. For each category $\mathcal{Q}_i$, describe how to imagine a cofunctor $\mathscr{C} \to \mathcal{Q}_i$ from an arbitrary category into it.
3. What cofunctors exist between the various $\mathcal{Q}_i$? ◊

Exercise 2.102. Let S be a set. Describe a way to visualize cofunctors from categories into the state category Sy^S. Feel free to focus on the case where S is a small finite set. Hint: use Proposition 2.87. ◊

Exercise 2.103.

1. Recall the star-shaped category $y^{\mathsf{n}+1} + \mathsf{n}y$ from Exercise 2.80. Describe cofunctors into it.
2. Describe cofunctors into Ay for a set A.
3. Describe cofunctors into $(\mathbb{N}, \leq)$.

4. Describe cofunctors into $(\mathbb{N}, \geq)$.
5. Let $y^4 + 2y^2 + y$ denote the commutative square category. List the cofunctors from it to the walking arrow category $y^2 + y$? There should be six or so. ◊

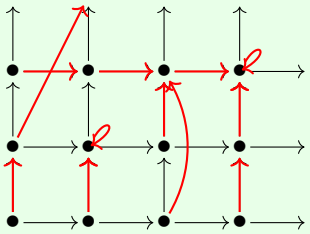
Example 2.104 (Objects aren't representable in $\mathbf{Cat}^\sharp$). For categories and ordinary functors, there is a category $\mathcal{T}$ that *represents objects*, in the sense that functors $\mathcal{T} \to \mathcal{C}$ are the same as objects in $\mathcal{C}$; indeed, take $\mathcal{T} = \boxed{\bullet}$ to be the terminal (one morphism) category.

This does not work for cofunctors, as we'll see in Exercise 2.105. The comonoid corresponding to $\mathcal{T}$ is y with its unique comonoid structure. Cofunctors $\mathcal{T} \nrightarrow \mathcal{C}$ are somewhat strange beasts: they can be identified with objects $c \in \mathcal{C}$ for which the codomain of every emanating morphism $c \to c'$ is $c' = c$ itself. The reason is the codomain condition (Definition 2.65, condition 2).

Exercise 2.105. We saw in Exercise 2.63 that $2y$ has a unique comonoid structure.

1. Show that for any category $\mathcal{T}$, there are $2^{\#\mathrm{Ob}(\mathcal{T})}$-many cofunctors $\mathcal{T} \nrightarrow 2y$.
2. Use the case of $\mathcal{C} := 2y$ to show that if a category $\mathcal{T}$ is going to represent objects as in Example 2.104 then $\mathcal{T}$ must have one object.
3. Now use a different $\mathcal{C}$ to show that if a category $\mathcal{T}$ is going to represent objects, it must have more than one object. ◊

Example 2.106 (Policies are co-representable). For a category $\mathcal{C}$, let's say that a *policy in* $\mathcal{C}$ is a choice, for each object $c \in \mathcal{C}$, of an emanating morphism $f : c \to c'$. For example, consider the category $(\mathbb{N}, \leq) \times (\mathbb{N}, \leq)$:



In red we have drawn a policy: every object has been assigned an emanating morphism to another object; there doesn't need to be any rhyme or reason to our choice.

For any category $\mathcal{C}$, the set of trajectories in $\mathcal{C}$ is in bijection with the set of cofunctors

$$\mathcal{C} \nrightarrow n$$

where $n = y^{\mathbb{N}}$ is the monoid of natural numbers under addition.

Exercise 2.107. At the end of Example 2.106 we said that a policy on $\mathcal{C}$ can be identified with a cofunctor $F\colon \mathcal{C} \nrightarrow \mathcal{N}$. But at first it appears that F includes more than just a policy: for every object $c \in \mathrm{Ob}(\mathcal{C})$ and natural number $n \in \mathbb{N}$, we have a morphism $F^\sharp_c(n)$ emanating from c. That's infinitely many emanating morphisms per object, whereas a policy seems to include only one emanating morphism per object.

Explain why looks are deceiving in this case: why is a policy on $\mathcal{C}$ the same as a cofunctor $\mathcal{C} \nrightarrow \mathcal{N}$? ◊

We will see later in Proposition 2.173 that the trajectories on a category form a monoid, and that this operation $\mathbf{Cat}^\sharp \to \mathbf{Mon}^{\mathrm{op}}$ is functorial and in fact an adjoint.

Exercise 2.108 (Continuous trajectories). Suppose we say that a continuous policy in $\mathcal{C}$ is a cofunctor $\mathcal{C} \nrightarrow \mathcal{R}$, where $\mathcal{R}$ is the monoid of real numbers under addition, considered as a category with one object.

Describe continuous trajectories in $\mathcal{C}$ using elementary terms, i.e. to someone who doesn't know what a cofunctor is and isn't yet ready to learn. ◊

Exercise 2.109. Let $\mathbb{R}/\mathbb{Z} \cong [0, 1)$ be the quotient of $\mathbb{R}$ by the $\mathbb{Z}$-action sending $(r, n) \mapsto r + n$. More down to earth, it's the set of real numbers between 0 and 1, including 0 but not 1.

1. Find a comonoid structure on $(\mathbb{R}/\mathbb{Z})y^{\mathbb{R}}$.
2. Is it a groupoid? ◊

Exercise 2.110.

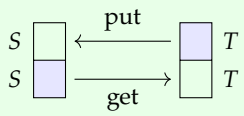
1. If two categories are isomorphic in **Cat**, does that imply they are isomorphic in $\mathbf{Cat}^\sharp$?
2. If so, prove it; if not, give a counterexample.
3. Is it true that for any two categories $\mathcal{C}, \mathcal{D}$, there is a bijection between the set of isomorphisms $\mathcal{C} \xrightarrow{\cong} \mathcal{D}$ in **Cat** and the set of isomorphisms $\mathcal{C} \overset{\cong}{\nrightarrow} \mathcal{D}$ between them in $\mathbf{Cat}^\sharp$?
4. If so, prove it; if not, give a counterexample. ◊

Very well behaved lenses In the functional programming community, there is an important notion of very well-behaved lenses. These turn out to be precisely the cofunctors between state categories. Since state categories Sy^S play an important role in our theory, we take a bit of time to consider cofunctors between them.

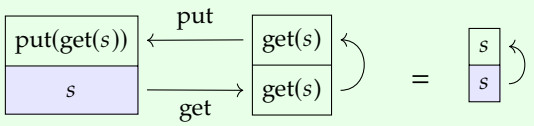
Example 2.111 (Very well-behaved lenses). Recall from Example 2.52 that for any set S, we have the "state" category with emanation polynomial Sy^S. What are the comonoid

morphisms—cofunctors—between different state categories?

First, such a comonoid morphism includes a morphism of polynomials $f : Sy^S \to Ty^T$; we'll use the standard terminology of "get" and "put":



Let's apply the unit-homomorphism property (2.84)



It says that put(get(s)) = s. This is typically called the get-put law.

We leave the comultiplication-homomorphism law to Exercise 2.112, where we will see that it specifies that get and put must satisfy two other properties, called the put-put and the put-get laws.

Exercise 2.112. Complete Example 2.111.

1. Write out the comultiplication law from (2.84) in terms of poly-boxes.
2. What set-theoretic equations are forced by the comultiplication law?
3. Can you see why they might be called put-put and put-get? ◊

Example 2.113 (Very well-behaved lenses are kinda boring). We saw in Exercise 2.112 that a comomonoid homomorphism (cofunctor) $Sy^S \nrightarrow Ty^T$ between state comonoids can be characterized as a pair of functions get: $S \to T$ and put: $S \times T \to S$ satisfying get-put, put-get, and put-put.

In fact, it turns out that this happens if and only if get is a product projection! For example, if the cardinalities $|S|$ and $|T|$ of S and T are finite and $|S|$ is not divisible by $|T|$, then there are no cofunctors $Sy^S \nrightarrow Ty^T$. A stringent condition, no? We'll explore it in Exercise 2.115 below.

Let's explain why cofunctors between state categories are just product projections. A product projection $A \times B \to A$ always has another factor (B); if every cofunctor between state categories is a product projection, what is the other factor†? It turns out that the other factor will be:

$$F \coloneqq \{f : T \to S \mid t \in T,\ \text{get}(f(t)) = t \text{ and } \text{put}(f(t), t) = f(t)\}.$$

In other words we will see that if (get, put) is a comonoid homomorphism then there

is a bijection $S \cong T \times F$ and that get: $S \to T$ is one of the projections. We will see that the converse is true in Exercise 2.114

So assume (get, put): $Sy^S \nrightarrow Ty^T$ is a comonoid homomorphism, in particular that it satisfies put-get, get-put, and put-put. We obtain a function $\pi : S \to T \times F$ given by

$$s \mapsto \big(\text{get}(s), t \mapsto \text{put}(s, t)\big)$$

and it is well-defined since for all $s \in S$ and $t, t' \in T$ we have $\text{get}(\text{put}(s, t)) = t$ by put-get and $\text{put}(\text{put}(s, t), t') = \text{put}(s, t')$ by put-put. We also obtain a function $\pi' : T \times F \to S$ given by

$$(t, f) \mapsto f(t).$$

The two functions π, π' are mutually inverse: the roundtrip on S is identity because $\text{put}(s, \text{get}(s)) = s$ by get-put; the roundtrip on $T \times F$ is identity because $\text{get}(f(t)) = t$ and $\text{put}(f(t), t) = f(t)$ by assumption on $f \in F$.

Exercise 2.114. Let S, T, F be sets and suppose given an isomorphism $\alpha : S \to T \times F$.

1. Show that there exists a very well behaved lens get: $S \to T$ and put: $S \times T \to S$.
2. Show that there exists a cofunctor between the state category on S and the state category on T.
3. Show that there exists a comonoid homomorphism $Sy^S \to Ty^T$ between the state comonoids. ◊

Exercise 2.115.

1. Suppose $|S| = 3$. How many cofunctors are there $Sy^S \to Sy^S$?
2. Suppose $|S| = 4$ and $|T| = 2$. How many cofunctors are there $Sy^S \nrightarrow Ty^T$? ◊

Exercise 2.116. Let S, T be sets and get: $S \to T$ and put: $S \times T \to S$ the parts of a very well behaved lens, i.e. a cofunctor $Sy^S \to Ty^T$ between state categories. Is it possible that put: $S \times T \to S$ is itself a product projection, i.e. sends $(s, t) \mapsto s$? ◊

When we get to cofree comonoids, we'll obtain a whole new class of cofunctors that are interesting to consider. But for now, we move on to more theory.

Some math about Cat$^\sharp$ We refer to morphisms between polynomial comonoids as cofunctors, again eliding the difference between comonoids in **Poly** and categories.

Proposition 2.117. The coproduct of polynomial comonoids agrees with the coproduct of categories. In particular, the initial comonoid is 0.

Proof. We refer the claim about 0 to Exercise 2.118.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be categories and $\mathfrak{c}_1, \mathfrak{c}_2$ their emanation polynomials, i.e. the carriers of the corresponding comonoids $\mathscr{C}_1$ and $\mathscr{C}_2$. We first notice that $\mathfrak{c} := \mathfrak{c}_1 + \mathfrak{c}_2$ is the carrier for the comonoid $\mathscr{C}$ corresponding to the sum category $\mathcal{C} := \mathcal{C}_1 + \mathcal{C}_2$. Indeed, the object-set of the sum is given by the sum of the object-sets

$$\mathrm{Ob}(\mathcal{C}_1 + \mathcal{C}_2) \cong \mathrm{Ob}(\mathcal{C}_1) + \mathrm{Ob}(\mathcal{C}_2),$$

and a morphism in $\mathcal{C}_1 + \mathcal{C}_2$ emanating from any such object is just a morphism in whichever of the categories $\mathcal{C}_1$ or $\mathcal{C}_2$ it is from.

It remains to show that $\mathscr{C}$ is the coproduct of $\mathscr{C}_1$ and $\mathscr{C}_2$ in $\mathbf{Cat}^\sharp$. Suppose given a comonoid $\mathscr{D}$ and comonoid homomorphisms (cofunctors) $f_1 \colon \mathscr{C}_1 \nrightarrow \mathscr{D}$ and $f_2 \colon \mathscr{C}_2 \nrightarrow \mathscr{D}$. Then for any object of $\mathscr{C}$ we have an associated object $f(c) \in \mathscr{D}$, given either by $f(c) := f_1(c)$ or by $f(c) := f_2(c)$ depending on whether $c \in \mathscr{C}_1$ or $c \in \mathscr{C}_2$. For any morphism m emanating from $f(c)$ we have a morphism $f^\sharp(m)$ emanating from c. It is easy to check that the cofunctor laws hold for f. Uniqueness of f given f_1, f_2 is also straightforward. □

Exercise 2.118.

1. Show that 0 is a comonoid.
2. Show that 0 is initial as a comonoid. ◊

Exercise 2.119. If $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ is a category, show there is an induced category structure on the polynomial $2\mathfrak{c}$. ◊

Exercise 2.120. Check that the terminal comonoid is y. ◊

Proposition 2.121. The category $\mathbf{Cat}^\sharp$ has products.

Proposition 2.122 (Porst)**.** The forgetful functor $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$ is comonadic.

Proof. The fact that a forgetful functor $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$ is comonadic if it has a right adjoint follows from Beck's monadicity theorem via an obvious generalization of an argument given by Paré in [**pare1969absolute**], as pointed out by Porst in [**porst2019colimits**]. □

Corollary 2.123. The category $\mathbf{Cat}^\sharp = \mathbf{Comon}(\mathbf{Poly})$ has all small colimits. They are created by the underlying polynomial functor $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$.

Proof. It is well known that a comonadic functor creates all colimits that exist in its codomain [**nlab:created-limit**]. By Theorem 1.212, the category **Poly** has all small colimits. □

Proposition 2.124. Let $\mathbf{Comon}(\mathbf{Poly})_{\mathrm{rep}}$ be the full subcategory of comonoids (c, ϵ, δ) in **Poly** for which the carrier $c = y^M$ is representable. Then there is an isomorphism of categories

$$\mathbf{Comon}(\mathbf{Poly})_{\mathrm{rep}} \cong \mathbf{Mon}^{\mathrm{op}}$$

where **Mon** is the category of monoids.

Proof. Let $\mathcal{C}$ be a category. It has only one object iff its emanation polynomial $\mathfrak{c}$ has only one position, i.e. $\mathfrak{c} \cong y^M$ for some $M \in \mathbf{Set}$, namely where M is the set of morphisms in $\mathcal{C}$. It remains to show that cofunctors between monoids are dual—opposite—to morphisms between monoids.

A cofunctor $f : y^M \to y^N$ involves a single function $f^\sharp : N \to M$ that must satisfy a law coming from unitality and one coming from composition, as in Definition 2.82. The result can now be checked by hand, or seen formally as follows. Each object in the two diagrams (2.83) is representable by Exercise 2.6. The Yoneda embedding $\mathbf{Set}^{\mathrm{op}} \to \mathbf{Poly}$ is fully faithful, so these two diagrams are equivalent to the unit and composition diagrams for monoid homomorphisms. □

Exercise 2.125. Let $\mathbf{Comon}(\mathbf{Poly})_{\mathrm{lin}}$ be the full subcategory of comonoids (c, ϵ, δ) in **Poly** for which the carrier $c = My$ is linear. Show that there is an isomorphism of categories

$$\mathbf{Comon}(\mathbf{Poly})_{\mathrm{lin}} \cong \mathbf{Set}. \qquad \lozenge$$

Proposition 2.126. The inclusion of linear comonoids into all comonoids has a left adjoint

$$\mathbf{Comon}(\mathbf{Poly}) \underset{Ay}{\overset{(\mathfrak{c} \triangleleft 1)y}{\underset{\Rightarrow}{\rightleftarrows}}} \mathbf{Comon}(\mathbf{Poly})_{\mathrm{lin}}$$

denoted by where they send a comonoid $(\mathfrak{c}, \epsilon, \delta)$ and a linear comonoid Ay.

Proof. We need to show that for any comonoid $(\mathfrak{c}, \epsilon, \delta)$ and set A, we have a natural isomorphism

$$\mathbf{Cat}^\sharp(\mathfrak{c}, Ay). \cong^? \mathbf{Cat}^\sharp((\mathfrak{c} \triangleleft 1)y, Ay)$$

But every morphism in Ay is an identity, so the result follows from the fact that every cofunctor must pass identities back to identities. □

A cofunctor (map of polynomial comonoids) is called *cartesian* if the underlying map $f : \mathfrak{c} \to \mathfrak{d}$ of polynomials is cartesian (i.e. for each position $i \in \mathfrak{c}(1)$, the map $f_i^\sharp : \mathfrak{d}[f_1(i)] \to \mathfrak{c}[i]$ is an isomorphism).

Proposition 2.127. Every cofunctor $f : \mathcal{C} \nrightarrow \mathcal{D}$ factors as a vertical morphism followed by a cartesian morphism

$$\mathcal{C} \overset{\text{vert}}{\nrightarrow} \mathcal{C}' \overset{\text{cart}}{\nrightarrow} \mathcal{D}.$$

Proof. A cofunctor $\mathcal{C} \nrightarrow \mathcal{D}$ is a map of polynomials $\mathfrak{c} \to \mathfrak{d}$ satisfying some properties, and any map of polynomials $f : \mathfrak{c} \to \mathfrak{d}$ can be factored as a vertical morphism followed by a cartesian morphism

$$\mathfrak{c} \xrightarrow{g} \mathfrak{c}' \xrightarrow{h} \mathfrak{d}.$$

For simplicity, assume $g_1 : \mathfrak{c}(1) \to \mathfrak{c}'(1)$ is identity (rather than merely isomorphism) on positions and similarly that for each $i \in \mathfrak{c}$ the map $h_i^\sharp : \mathfrak{c}'[i] \to \mathfrak{d}[h_1(i)]$ is identity (rather than merely isomorphism) on directions.

It suffices to show that the intermediate object $\mathfrak{c}'$ can be endowed with the structure of a category such that g and h are cofunctors. Given an object $i \in \mathfrak{c}'(1)$, assign its identity to be the identity on $h_1(i) = f(i)$; then both g and h preserve identities because f does. Given an emanating morphism $m \in \mathfrak{c}'[i] = \mathfrak{d}[f(i)]$, assign its codomain to be $\mathrm{cod}(m) \coloneqq \mathrm{cod}(f_i^\sharp(m))$, and given an emanating morphism $m' \in \mathfrak{c}'[\mathrm{cod}(m)]$, assign the composite $m \fatsemi m'$ in $\mathfrak{c}'$ to be $m \fatsemi m'$ in $\mathfrak{d}$. In Exercise 2.128 we will check that with these definitions, $\mathfrak{c}'$ is a category and both g and h are cofunctors. □

Exercise 2.128. We will complete the proof of Proposition 2.127, using the same notation.

1. Show that composition is associative and unital in $\mathfrak{c}'$.
2. Show that g preserves codomains.
3. Show that g preserves compositions.
4. Show that h preserves codomains.
5. Show that h preserves compositions. ◊

Proposition 2.129. The wide subcategory of cartesian maps in $\mathbf{Cat}^\sharp$ is isomorphic to the category of wide subcategory of discrete opfibrations in **Cat**.

Proof. Suppose that $\mathcal{C}$ and $\mathcal{D}$ are categories. Both a functor and a cofunctor between them involve a map on objects, say $f : \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{D})$. For any object $c \in \mathrm{Ob}(\mathcal{C})$, a functor gives a function, say $f_\sharp : \mathcal{C}[c] \to \mathcal{D}[f(c)]$ whereas a cofunctor gives a function

$f^\sharp\colon \mathcal{D}[f(c)] \to \mathcal{C}[c]$. The cofunctor is cartesian iff $f^\sharp$ is an iso, and the functor is a discrete opfibration iff $f_\sharp$ is an iso. We thus transform our functor into a cofunctor (or vice versa) by taking the inverse function on morphisms. It is easy to check that this inverse appropriately preserves identities, codomains, and compositions. □

Proposition 2.130. The wide subcategory of vertical maps in $\mathbf{Cat}^\sharp$ is isomorphic to the opposite of the wide subcategory bijective-on-objects maps in **Cat**:

$$\mathbf{Cat}^\sharp_{\text{vert}} \cong (\mathbf{Cat}_{\text{boo}})^{\text{op}}.$$

Proof. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. Given a vertical cofunctor $F\colon \mathcal{C} \nrightarrow \mathcal{D}$, we have a bijection $F_1\colon \text{Ob}(\mathcal{C}) \to \text{Ob}(\mathcal{D})$; let G_1 be its inverse. We define a functor $G\colon \mathcal{D} \to \mathcal{C}$ on objects by G_1 and, for any $f\colon d \to d'$ in $\mathcal{D}$ we define $G(f) \coloneqq F^\sharp_{G_1(d)}$. It has the correct codomain: $\text{cod}(G(f)) = G_1(F_1(\text{cod}(G(f))) = G_1(\text{cod}\, f)$. And it sends identities and compositions to identities and compositions by the laws of cofunctors.

The construction of a vertical cofunctor from a bijective-on-objects functor is analogous, and it is easy to check that the two constructions are inverses. □

Exercise 2.131. Let S be a set and consider the state category $\mathcal{S} \coloneqq (Sy^S, \epsilon, \delta)$. Use Proposition 2.130 to show that categories $\mathcal{C}$ equipped with a vertical cofunctor $\mathcal{S} \to \mathcal{C}$ can be identified with categories whose set of objects is S. ◊

Exercise 2.132. Consider the categories $\mathcal{C} \coloneqq \boxed{\bullet \rightrightarrows \bullet}$ and $\mathcal{D} \coloneqq \boxed{\bullet \to \bullet}$. There is a unique bijective-on-objects (boo) functor $F\colon \mathcal{C} \to \mathcal{D}$ and two boo functors $G_1, G_2\colon \mathcal{D} \to \mathcal{C}$.

1. Write down the morphism $\mathfrak{d} \to \mathfrak{c}$ of emanation polynomials underlying F.
2. Write down the morphism $\mathfrak{c} \to \mathfrak{d}$ of emanation polynomials underlying either G_1 or G_2. ◊

Dirichlet monoidal product on $\mathbf{Cat}^\sharp$. The usual product of categories gives a monoidal operation on comonoids too, even though it is not a product in $\mathbf{Cat}^\sharp$. The carrier polynomial of the product is the $\otimes$-product of the carrier polynomials.

Proposition 2.133. The Dirichlet monoidal product $(y, \otimes)$ on **Poly** extends to a monoidal structure $(y, \otimes)$ on $\mathbf{Cat}^\sharp$, such that the functor $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is strong monoidal with respect to $\otimes$. The Dirichlet product of two categories is their product in **Cat**.

Proof. Let $\mathcal{C}, \mathcal{D} \in \mathbf{Cat}^\sharp$ be categories with emanation polynomials $\mathfrak{c}, \mathfrak{d} \in \mathbf{Poly}$. The emanation polynomial of $\mathcal{C} \otimes \mathcal{D}$ is defined to be $\mathfrak{c} \otimes \mathfrak{d}$. A position in it is a pair (c, d)

of objects, one from $\mathcal{C}$ and one from $\mathcal{D}$; a direction there is a pair (f, g) of a morphism emanating from c and one emanating from d.

We define $\epsilon_{\mathcal{C}\otimes\mathcal{D}} : \mathfrak{c} \otimes \mathfrak{d} \to y$ as

$$\mathfrak{c} \otimes \mathfrak{d} \xrightarrow{\epsilon_{\mathfrak{C}} \otimes \epsilon_{\mathfrak{D}}} y \otimes y \cong y.$$

This says that the identity at (c, d) is the pair of identities.

We define $\delta_{\mathcal{C}\otimes\mathcal{D}} : (\mathfrak{c} \otimes \mathfrak{d}) \to (\mathfrak{c} \otimes \mathfrak{d}) \triangleleft (\mathfrak{c} \otimes \mathfrak{d})$ using the duoidal property:

$$\mathfrak{c} \otimes \mathfrak{d} \xrightarrow{\delta_{\mathfrak{c}} \otimes \delta_{\mathfrak{d}}} (\mathfrak{c} \triangleleft \mathfrak{c}) \otimes (\mathfrak{d} \triangleleft \mathfrak{d}) \to (\mathfrak{c} \otimes \mathfrak{d}) \triangleleft (\mathfrak{c} \otimes \mathfrak{d}).$$

One can check that this says that codomains and composition are defined coordinate-wise, and that $(\mathfrak{c} \otimes \mathfrak{d}, \epsilon_{\mathfrak{c}\otimes\mathfrak{d}}, \delta_{\mathfrak{c}\otimes\mathfrak{d}})$ forms a comonoid. One can also check that this is functorial in $\mathcal{C}, \mathcal{D} \in \mathbf{Cat}^\sharp$. See Exercise 2.134. □

Exercise 2.134. We complete the proof of Proposition 2.133.

1. Show that $(\mathfrak{c} \otimes \mathfrak{d}, \epsilon_{\mathfrak{c}\otimes\mathfrak{d}}, \delta_{\mathfrak{c}\otimes\mathfrak{d}})$, as described in Proposition 2.133, forms a comonoid.
2. Check that the construction $(\mathcal{C}, \mathcal{D}) \mapsto \mathcal{C} \otimes \mathcal{D}$ is functorial in $\mathcal{C}, \mathcal{D} \in \mathbf{Cat}^\sharp$. ◊

2.3 Cofree comonoids

2.3.1 Introduction: Cofree comonoids and dynamical systems

We can now return to dynamical systems. Recall that if $p \in \mathbf{Poly}$ is a polynomial, then a dynamical system with interface p consists of a set S and a map of polynomials $f : Sy^S \to p$. We think of positions in p kind of like outputs—others can observe your position—but also as determining the set of inputs—directions—that could be input next.

The way this map f leads to something that seems to "go on repeatedly" is that $\mathfrak{s} \coloneqq Sy^S$ is a comonoid, so we have maps $\mathfrak{s} \xrightarrow{\delta} \mathfrak{s}^{\triangleleft n} \xrightarrow{f^{\triangleleft n}} p^{\triangleleft n}$ for all n. This says that given any initial position in S, we automatically get a position in p, and for every direction there, another position in p, and for every direction there, another position in p, and so on n times. This is the dynamics.

So the above all works because we have a polynomial map $Sy^S \to p$, where Sy^S is the underlying polynomial of a polynomial comonad. Since "underlying polynomial" is a functor $U : \mathbf{Cat}^\sharp \to \mathbf{Poly}$, a seasoned category theorist might be tempted to ask, is there an adjunction

$$\mathbf{Poly}(U\mathscr{C}, p) \cong \mathbf{Cat}^\sharp(\mathscr{C}, \mathscr{T}_p),$$

for some functor $\mathscr{T}$: $\mathbf{Poly} \to \mathbf{Cat}^\sharp$? In fact, there is; we refer to $\mathscr{T}_p$ as the *cofree comonoid* on p, or more descriptively, the *category of p-trees.*

Cofree comonoids in **Poly** are beautiful objects, both in their visualizable structure as a category and in the metaphors we can make about them. They allow us to replace

the interface of a dynamical system with a category and get access to a rich theory that exists there.

Theorem 2.135 (Cofree comonoid)**.** The forgetful functor **Comon**(**Poly**) → **Poly** has a right adjoint

$$\mathbf{Cat}^\sharp \underset{\mathscr{T}_p}{\overset{\mathfrak{c}}{\rightleftarrows}} \mathbf{Poly}$$

where the functors have been named by where they send $(\mathfrak{c}, \epsilon, \delta) \in \mathbf{Cat}^\sharp$ and $p \in \mathbf{Poly}$ respectively.

This will be proved as **??**.

2.3.2 Cofree comonoids as trees

Definition 2.136 (Cofree comonoid)**.** Let $p \in \mathbf{Poly}$ be a polynomial. The comonoid $\mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$ as in Theorem 2.159 is called the *cofree comonoid on* p or informally the category of *(possibly infinite) p-trees*.

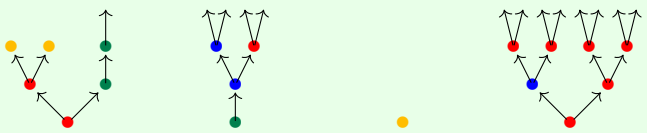
An object $t \in \mathsf{tree}_p(1)$ is a called a *(possibly infinite) tree in* p. Given such an object t in the category, an emanating morphism $n \in \mathsf{tree}_p[t]$ is called a *path from root*.

The terminology of Definition 2.136 is alluding to a specific way we like to imagine the comonoid $\mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$, namely in terms of trees. To every polynomial p, we will associate a new polynomial tree_p whose positions are (possibly infinite) p-trees. To choose such tree we first choose its root to be some position $i \in p(1)$. Then for every direction $d \in p[i]$ there, we choose another position, and for every direction from each of those we choose another position, and so on indefinitely.

So a position in tree_p is one of these trees. Such a tree may end, namely if every one of the top-level positions have no directions, but often it will not end. Given such a tree, say t, a direction $d \in \mathsf{tree}_p[t]$ there is simply a path from the root to some node in the tree.

We'll explain the counit root and the comultiplication focus after going through an example.

Example 2.137. Let $p := \{\bullet, \bullet\}y^2 + \bullet y + \bullet$. Here are four trees in p:



(2.138)

They all represent elements of $p^{\triangleleft 3}$, but only the third one—the single yellow dot—would count as an element of tree_p.

Indeed, in Definition 2.136, when we speak of a (possibly infinite) tree in p, we mean at tree for which each node is a position in p with each of its emanating directions filled by another position in p, and so on. Since three of the four trees shown in (2.138) have open leaves—arrows emanating from the top—these trees are not elements of tree_p. However, each of them could be extended to an actual element of tree_p by continually filling in each open leaf with another position of p.

Let's imagine some actual elements of tree_p:

- The binary tree that's "all red all the time".
- The binary tree where odd layers are red and even layers are blue.
- The tree where all the nodes are red, except for the right-most branch, which is always green.[a]
- Any finite tree, where every branch terminates in a yellow dot.
- Completely random: for the root, randomly choose either red, blue, green, or yellow, and at every leaf, loop back to the beginning, i.e. randomly choose either red, blue, green, or yellow, etc.

These are the positions in the polynomial tree_p that underlies the cofree comonoid on p. There are uncountably many positions in $\mathscr{T}_p$, at least for this particular p—in fact, even $\mathscr{T}_{2y}$ has $2^{\mathbb{N}}$-many positions—but only finitely many can be described in any finite language like English. Thus most of the elements of $\mathscr{T}_p$ cannot be described.

[a] Note that branches are actually unordered, so it's technically wrong to think of it as a line of green up the *right side*. Instead, it's just a line of green going up the tree forever.

Exercise 2.139.

1. Interpret each of the five tree examples imagined in Example 2.137 by drawing three or four layers (your choice) of it.

For each of the following polynomials p, describe the set of trees (positions) in tree_p.

2. $p = 1$. (What is the set tree_p of p-trees?)
3. $p = 2$.
4. $p = y$.
5. $p = y^2$.
6. $p = 2y$.
7. $p = y + 1$.
8. $p = By^A$ for some sets $A, B \in \mathbf{Set}$. ◊

Now that we've explained the underlying polynomial tree_p of the cofree comonoid $\mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$, we just need to explain how identities, codomains, and composition work, i.e. we just need to give the counit map $\mathrm{root} : \mathsf{tree}_p \to y$ and the comultiplication map $\mathrm{focus} : \mathsf{tree}_p \to \mathsf{tree}_p \triangleleft \mathsf{tree}_p$.

Again, the objects in the category $\mathscr{T}_p$ are p-trees, and a morphism emanating from such a tree t is a path from its root r to some node. The map root, applied to t, returns t's root r, or more precisely the path from r to itself. The map focus, applied to t, first

needs to give a codomain (tree) to every path from the root to some other node n. It is just the subtree of t whose root is n: the tree of all nodes starting at n. Now, given a path from the root of that tree (namely n) to another node, say n', we need to give a path from r to n'; we take it to be the composite of the path $r \to n$ and the path $n \to n'$.

Exercise 2.140. Let $p := \{\bullet, \bullet\}y^2 + \bullet y + \bullet$ as in Example 2.137.

1. Choose an object $t \in \mathsf{tree}_p$, i.e. a tree in p, and draw a finite approximation of it (say four layers).
2. What is the identity morphism at t?
3. Choose a nonidentity morphism f emanating from t and draw it.
4. What is the codomain of f? Draw a finite approximation of it.
5. Choose a morphism emanating from the codomain of f and draw it.
6. What is the composite of your two morphisms? Draw it on t. ◊

Example 2.141. Let's take $p := 1$. An element in $\mathsf{tree}_p(1)$ is given by choosing an element $i \in p(1)$ and filling each of its direction $p[i]$ with another element of $p(1)$, and so on. But there is only one element of $p(1)$ and it has no directions. So tree_1 has only one position, and the only emanating morphism there is the identity. In other words, $\mathsf{tree}_1 = y$.

Since y has a unique comonoid structure, we've described the cofree comonoid $\mathscr{T}_(1)$. It is a single tree consisting of a single node, and the only outgoing morphism is the identity on that node.

Exercise 2.142. Let A be a set.

1. What is tree_A?
2. How is it given the structure of a category? ◊

Example 2.143. Let $p := y$. An element in $\mathsf{tree}_p(1)$ is given by choosing an element $i \in p(1)$ and filling each of its direction $p[i]$ with another element of $p(1)$, and so on. There is only one way to do this, i.e. there is only one such tree, namely $t := \boxed{\bullet \to \bullet \to \bullet \to \cdots}$.

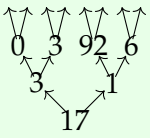
So tree_p has a single position, namely t. That position has an emanating morphism for each path out of the root, so it has $\mathbb{N}$-many emanating morphisms: one for every length. Hence $\mathsf{tree}_y = y^{\mathbb{N}}$.

Of course the codomain of each morphism emanating from t is again t: that's the only object. The composite of two paths, one of length m and one of length n is $m + n$. Hence we see that the category $\mathscr{T}(y)$ is the monoid $(\mathbb{N}, 0, +)$ considered as a category with one object.

Exercise 2.144. Let A be a set.

1. What is tree_{Ay}?
2. How is it given the structure of a category? ◊

Example 2.145. Let $p := \mathbb{N}y^2$. An element of tree_p might start like this:



Any element of tree_p goes on forever: it's an infinite binary tree. At each node it has a choice of some natural number, since $\mathbb{N} = p(1)$ is the set of positions in p.

So such trees are the objects of the category $\mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$. A morphism emanating from a tree t is a path from its root to another node, which is an element of $\mathsf{List}(2)$, i.e. a finite list of choices in 2, which you can think of as a finite sequence of left/right choices. The codomain is whatever tree this path ends up on.

So the emanation polynomial of $\mathscr{T}_p$ is

$$\mathsf{tree}_p \cong \mathbb{N}^{\mathsf{List}(2)} y^{\mathsf{List}(2)}$$

with identities given by the empty list. An object $t \in \mathsf{tree}_p(1)$ is a function $t : \mathsf{List}(2) \to \mathbb{N}$, a way to put a natural number at every node of the infinite binary tree. An emanating morphism $\ell \in \mathsf{List}(2)$ is just a path from the root to another node, and its codomain is the other node. Formally it is the function $t' : \mathsf{List}(2) \to \mathbb{N}$ given by $t'(\ell') := t(\ell : \ell')$, where $\ell : \ell'$ is the concatenation of these lists. Composition of morphisms is also given by concatenation of the corresponding lists.

Exercise 2.146. Let $p := By^A$ for sets $A, B \in \mathbf{Set}$.

1. Describe the objects of the cofree category $\mathscr{T}_p$.
2. For a given such object, describe the emanating morphisms.
3. Describe how to take the codomain of a morphism. ◊

Example 2.147 ($\mathscr{T}_{Ay}$ for linear polynomials). Let $A \in \mathbf{Set}$ be a set. The cofree comonoid $\mathscr{T}_{Ay}$ on the associated linear polynomial has as emanation polynomial $\mathsf{tree}_{Ay} \cong (Ay)^{\mathbb{N}}$. Its objects are A-streams. For each stream $t : \mathbb{N} \to A$, an emanating morphism is just an element $n \in \mathbb{N}$. The identity is $0 \in \mathbb{N}$, the codomain of n is the composite function $\mathbb{N} \xrightarrow{+n} \mathbb{N} \xrightarrow{t} A$, and if we denote it by $n : t \to (t + n)$ then the composite of morphisms

n, n' is $(n + n') \colon t \to (t + n + n')$.

We first saw this category in Example 2.64

Exercise 2.148. Let $p \coloneqq y + 1$.

1. Describe the objects of the cofree category $\mathscr{T}_p$.
2. For a given such object, describe the emanating morphisms.
3. Describe how to take the codomain of a morphism. ◊

Exercise 2.149. Let $p \coloneqq \{a, b, c, \ldots, z, \sqcup\}y + 1$.

1. Describe the objects of the cofree category $\mathscr{T}_p$, and draw one.
2. For a given such object, describe the emanating morphisms.
3. Describe how to take the codomain of a morphism. ◊

Decision trees. When you talk about your future, what exactly might you be talking about? In some sense you can make choices that change what will happen to you, but in another sense it's as though for each such choice there is something beyond your control that makes a new situation for you. You're constantly in the position of needing to make a choice, but its results are beyond your control.

This is very much how positions $t \in \mathscr{T}_p$ look. Such a position is a decision tree: at each stage (node), you have an element $i \in p(1)$, which we've been calling a decision. It has $p[i]$-many options, each of which, say $d \in p[i]$ results in a new node $\mathrm{cod}(d)$ of the tree.

So a position t is like a future: it is a current decision, and for every option there, a new decision tree. It's all the decisions you could possibly make, and for each actual choice, it's a new future. A direction at t is just a choice of finite path through the tree: a sequence of choices.

Exercise 2.150. If someone says that they understand a future to be a decision tree $t \in \mathscr{T}_p$, explain in your own words how they're thinking about the term "future". How does it agree or disagree with your own intuition about what a "future" is? ◊

Exercise 2.151. Let G be a finite directed graph, and let $\mathbf{Fr}(G)$ be the associated free category.

1. Construct a cofunctor $\mathbf{Fr}(G) \nrightarrow \mathscr{T}_{1+y+y^2+y^3+\cdots}$.
2. Would you say it associates to each node in G its "future" decision tree? ◊

2.3.3 Formal construction of $\mathscr{T}_p$

We will sketch how one can formally construct $\mathscr{T}_p$ from p. The first step is called copointing, and it's pretty easy: just multiply p by y. It adds a kind of "default" direction to each position in p.

Copointing.

Definition 2.152 (Copointed polynomial)**.** A *copointed polynomial* is a pair (p, ϵ), where $p \in \mathbf{Poly}$ is a polynomial and $\epsilon : p \to y$ is a morphism in **Poly**.

A *morphism* of copointed polynomials $f : (p, \epsilon) \to (p', \epsilon')$ is a morphism $f : p \to p'$ such that $\epsilon = f \fatsemi \epsilon'$.

Comonoids in **Poly** are triples (p, ϵ, δ), with (p, ϵ) a copointed polynomial, so there are forgetful functors

$$\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Cpt}(\mathbf{Poly}) \to \mathbf{Poly}.$$

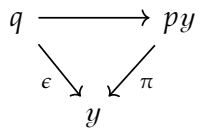
We want to find the right adjoint to the composite—that's the functor $\mathscr{T}_- : \mathbf{Poly} \to \mathbf{Comon}(\mathbf{Poly})$—and we will obtain it in two steps.

Proposition 2.153. For any polynomial p, the polynomial py is naturally copointed by the projection to y, and the functor sending $p \mapsto py$ is right adjoint to the forgetful functor

$$\mathbf{Cpt}(\mathbf{Poly}) \underset{q}{\overset{py}{\underset{\longleftarrow}{\overset{\longrightarrow}{\Rightarrow}}}} \mathbf{Poly} \ ,$$

where the functors are named by where they send $p \in \mathbf{Poly}$ and $(q, \epsilon) \in \mathbf{Cpt}(\mathbf{Poly})$.

Proof. Clearly the product $py \cong p \times y$ is copointed by the projection map, call it $\pi : py \to y$, and the map $p \mapsto py$ is functorial in p. For any copointed polynomial $q \xrightarrow{\epsilon} y$, there is an obvious bijection between morphisms of polynomials $q \to p$ and commutative triangles



natural in both q and p. This completes the proof. □

Exercise 2.154. Show that the copointing functor is essentially surjective. That is, every polynomial p equipped with a map $\epsilon : p \to y$ is isomorphic to one of the form $p'y$ (equipped with the projection $p'y \to y$). ◊

The reader might not remember any sort of copointing showing up in the tree description of $\mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$. Indeed, it was hidden in the fact that we allowed for trivial paths in the tree (e.g. the path from the root to itself). But we'll get to that.

The copointing $p \mapsto py$ just adds an extra direction to each position; we can denote this extra direction with an =, as we did in Example 2.61. So for example if $p = y^3 + y$, drawn as left, then $py \cong y^4 + y^2$ can be drawn as right:

p $\qquad$ py

It just adds a default direction to each position. A copointed map from (q, ϵ) to (py, π) must pass the default direction back to the default direction in q, but leaves the other directions in p to go wherever they want to.

Example 2.155 (Slowing down dynamical systems). Given a dynamical system $f : Sy^S \to p$, we automatically get a map $Sy^S \to py$ to the cofree pointing. We called this "adding a pause button" in Example 1.132. Thus we can take any dynamical system and replace it with one whose interface is copointed.

We can use a copointed interface to slow down a dynamical system, in a kind of inverse to how we sped up dynamical systems in (2.59). There we took a dynamical system f with interface p and produced one and produced one with interface $p^{\triangleleft k}$. Here we will take one with interface $p^{\triangleleft k}$ and produce one with interface p.

To do this, we need p to be copointed, i.e. we need to have in hand a map $\epsilon : p \to y$, and as we saw above that we can always assume that. Now for any $k \in \mathbb{N}$ we have k-many maps $p^{\triangleleft k} \to p$. For example, if $k = 3$, we have

$$\{\epsilon \triangleleft \epsilon \triangleleft p, \epsilon \triangleleft p \triangleleft \epsilon, p \triangleleft \epsilon \triangleleft \epsilon\} \subseteq \mathbf{Poly}(p^{\triangleleft 3}, p)$$

So given a dynamical system $Sy^S \to p^{\triangleleft k}$, which outputs as its position a whole k-fold strategy at one time and which takes as input sequences of k-many inputs, we can feed it one input and $k-1$ pauses. This is what you get when you compose $Sy^S \to p^{\triangleleft k} \to p$.

Given a dynamical system $f : Sy^S \to p$, where p is copointed and f preserves the copoint, we could speed it up as before to get $Sy^S \to p^{\triangleleft k}$ and then slow it down to get $Sy^S \to p$, and we get back f. So slowing down is a retract of speeding up in this sense.

Exercise 2.156.

1. Show that there is a monoidal structure $(y, \otimes)$ on $\mathbf{Cpt}(\mathbf{Poly})$ such that the forgetful functor $U : \mathbf{Cpt}(\mathbf{Poly}) \to \mathbf{Poly}$ is strong monoidal.

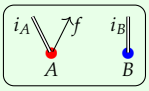
2. Show that this monoidal structure is closed, i.e. that there is an internal hom $[-,-]$ on **Cpt**(**Poly**). Hint: you should have $U([py, qy]_{\mathbf{Cpt}}) \cong [py, q]_{\mathbf{Poly}} y$. ◊

Constructing the cofree comonoid. It remains to show that we can functorially take any copointed polynomial (q, ϵ) and return a comonoid, and that this construction is right adjoint to the forgetful functor. From the description in Section 2.3.2, we know the cofree comonoid is supposed to have something to do with infinite trees. And we know that the set of height-n p-trees is given by $p^{\triangleleft n}$. So we might think we can somehow take a limit of these height-n trees for various n.

The problem is there's no obvious maps between $p^{\triangleleft n}$ and $p^{\triangleleft n+1}$. Luckily, the copointing fixes that problem. Given $\epsilon : q \to y$, we have two maps $q \triangleleft q \rightrightarrows q$, namely $q \triangleleft \epsilon$ and $\epsilon \triangleleft q$.

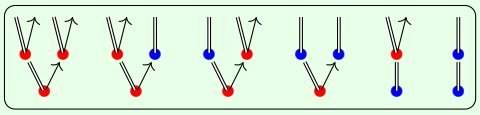
Example 2.157. Suppose $q = \{A\}y^{\{i_A, f\}} + \{B\}y^{i_B}$ with copointing ϵ selecting i_A and i_B:

$$q \coloneqq$$



Then $q \triangleleft q$ looks as follows

$$q \triangleleft q =$$



How can we picture the maps $(q \triangleleft \epsilon), (\epsilon \triangleleft q) : q \triangleleft q \to q$?

The map $q \triangleleft \epsilon$ takes each position of $q \triangleleft q$ to whatever is on the bottom layer: it takes the first four to A and the last two to B. It passes back directions using the defaults (i_A and i_B) on the top layer.

The map $\epsilon \triangleleft q$ uses the defaults on the bottom layer instead. Every position in $q \triangleleft q$ has a default direction, and the corolla sitting there in the top layer is where $\epsilon \triangleleft q$ sends it, with identity on directions.

Indeed, for every n, there are $(n+1)$-many morphisms $q^{\triangleleft n+1} \to q^{\triangleleft n}$, so we have a diagram

$$y \xleftarrow{\epsilon} q \underset{q \triangleleft \epsilon}{\overset{\epsilon \triangleleft q}{\leftleftarrows}} q \triangleleft q \underset{q \triangleleft q \triangleleft \epsilon}{\overset{\epsilon \triangleleft q \triangleleft q}{\lefttriplearrows}} q^{\triangleleft 3} \quad \cdots \tag{2.158}$$

The cofree comonoid is given by the limit of this diagram.

Let's denote the shape of this diagram by Δ_+: its objects are finite ordered sets—including the empty set—and its morphisms are order-preserving injections. For any copointed polynomial $q \xrightarrow{\epsilon} y$, we get a diagram $Q : \Delta_+ \to \mathbf{Poly}$ as above, and this is functorial in q.

Theorem 2.159. For any copointed polynomial $q \xrightarrow{\epsilon} y$, let $\bar{q}$ denote the limit of $Q\colon \Delta_+ \to$ **Poly**. It naturally has the structure of a comonoid $(\bar{q}, \epsilon, \delta)$, and this construction is right adjoint to the forgetful functor

$$\textbf{Comon}(\textbf{Poly}) \underset{\bar{q}}{\overset{U}{\underset{\Leftarrow}{\overset{\longrightarrow}{\longleftarrow}}}} \textbf{Cpt}(\textbf{Poly}) \ .$$

Proof sketch. We first give $\bar{q}$ the structure of a comonoid. Since $\bar{q}$ is the limit of (2.158), the inclusion the inclusion $\{0\} \to \Delta_+$ induces a projection map $\bar{q} \to y$, which we again call ϵ. Since $\triangleleft$ commutes with connected limits in both variables and Δ_+ is connected, we have that $\bar{q} \triangleleft \bar{q}$ is the limit of the following $\Delta_+ \times \Delta_+$-shaped diagram:

$$\bar{q} \triangleleft \bar{q} \cong \lim \left(\begin{array}{ccccccccc}
q^{\triangleleft(0+0)} & \longleftarrow & q^{\triangleleft(0+1)} & \Lleftarrow & q^{\triangleleft(0+2)} & \Lleftarrow & q^{\triangleleft(0+3)} & \Lleftarrow & \cdots \\
\uparrow & & \uparrow & & \uparrow & & \uparrow & & \\
q^{\triangleleft(1+0)} & \longleftarrow & q^{\triangleleft(1+1)} & \Lleftarrow & q^{\triangleleft(1+2)} & \Lleftarrow & q^{\triangleleft(1+3)} & \Lleftarrow & \cdots \\
\upuparrows & & \upuparrows & & \upuparrows & & \upuparrows & & \\
q^{\triangleleft(2+0)} & \longleftarrow & q^{\triangleleft(2+1)} & \Lleftarrow & q^{\triangleleft(2+2)} & \Lleftarrow & q^{\triangleleft(2+3)} & \Lleftarrow & \cdots \\
\Uparrow & & \Uparrow & & \Uparrow & & \Uparrow & & \\
q^{\triangleleft(3+0)} & \longleftarrow & q^{\triangleleft(3+1)} & \Lleftarrow & q^{\triangleleft(3+2)} & \Lleftarrow & q^{\triangleleft(3+3)} & \Lleftarrow & \cdots \\
\Uparrow & & \Uparrow & & \Uparrow & & \Uparrow & & \\
\vdots & & \vdots & & \vdots & & \vdots & & \ddots
\end{array}\right)$$

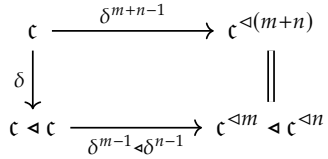
There is a commutative diagram in **Cat**

$$\begin{array}{ccc}
\Delta_+ \times \Delta_+ & \xrightarrow{\quad + \quad} & \Delta_+ \\
& {\scriptstyle (m_1,m_2) \mapsto q^{\triangleleft(m_1+m_2)}} \searrow \quad \swarrow {\scriptstyle n \mapsto q^{\triangleleft n}} & \\
& \textbf{Poly} &
\end{array} \tag{2.160}$$

which induces a map (in the opposite direction) between their limits $\delta\colon \bar{q} \to \bar{q} \triangleleft \bar{q}$, which we take to be the comultiplication. Appending (2.160) with the inclusion $\{0\} \times \Delta_+ \to \Delta_+ \times \Delta_+$, etc., it is easy to see that $(\bar{q}, \epsilon, \delta)$ satisfies the axioms of a comonoid.

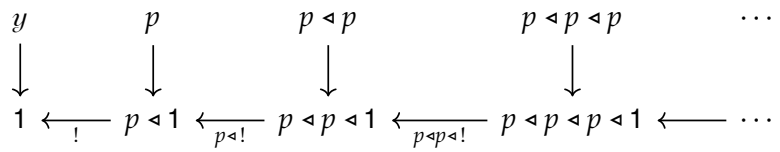
We sketch the proof that this construction is right adjoint to the forgetful functor. For any copointed polynomial (q, ϵ), there is a counit map $\bar{q} \to q$, given by the obvious projection of the limit (2.158). Given a comonoid $(\mathfrak{c}, \epsilon, \delta)$, there is a morphism $\mathfrak{c} \to \bar{\mathfrak{c}}$ induced by the maps $\delta^{n-1}\colon \mathfrak{c} \to \mathfrak{c}^{\triangleleft n}$ from Example 2.51. It is easy to check that these commute with the ϵ's in the diagram. To see that $\mathfrak{c} \to \bar{\mathfrak{c}}$ extends to a morphism of

comonoids amounts to checking that the diagram

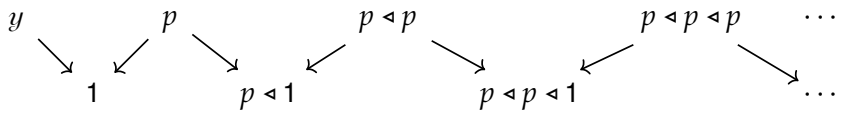


commutes for any $m, n \in \mathbb{N}$. Both triangle equations are straightforward. □

Remark 2.161. The construction of the cofree comonoid from a copointed endofunctor in the proof of Theorem 2.159 is fairly standard; see **[lack2010note]**. Nelson Niu has also constructed the cofree comonoid on a polynomial using a different limit diagram



in terms of the original polynomial p, rather than from its copointing py; that is, this construction is right adjoint to $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$. One could also construct this right adjoint using the following limit, again applied to the original polynomial p:



We record the following proposition here; it will be useful in Corollary 3.75.

Proposition 2.162. If $f : p \to q$ is a Cartesian map of polynomials, then $\mathsf{tree}_f : \mathsf{tree}_p \to \mathsf{tree}_q$ is a Cartesian cofunctor. That is, for each tree $t \in \mathsf{tree}_p(1)$ the function $\mathsf{tree}_f^\sharp : \mathsf{tree}_p[t] \xrightarrow{\cong} \mathsf{tree}_q[\mathsf{tree}_f(t)]$ is a bijection.

Proof. ** □

Proposition 2.163. The cofree comonoid functor is lax monoidal; in particular, we have maps

$$y \to \mathscr{T}_y \qquad \text{and} \qquad \mathscr{T}_p \otimes \mathscr{T}_q \to \mathscr{T}_{p \otimes q}$$

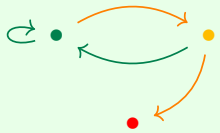
for any $p, q \in \mathbf{Poly}$.

2.3.4 Consequences of adjointness

Recall from Definition 1.127 that a dependent system (or generalized Moore machine) is a map of polynomials $f : Sy^S \to p$. Here S is called the set of states and p is the interface.

But now we know that Sy^S is secretly the underlying polynomial of a comonoid. This means that f has a mate, i.e. a corresponding cofunctor $Sy^S \nrightarrow \mathscr{T}_p$ to the category of p-trees. How does that work?

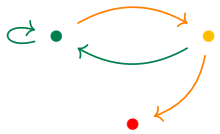
Example 2.164. Let $S := \{\bullet, \bullet, \bullet\}$ and $p := y^2 + y$, and consider the dynamical system $f : Sy^S \to p$ from Exercise 1.129, depicted here again for your convenience:



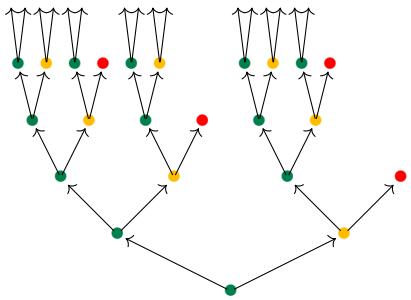
The polynomial map f is supposed to induce a cofunctor $F : Sy^S \nrightarrow \mathscr{T}_p$ from the state category on S to the category of p-trees. Thus to each state $s \in S$, we need to associate a tree; which should it be?

Exercise 2.165. Consider the walking arrow category $\mathcal{W} = \boxed{\bullet \to \bullet}$. Draw the cofunctor $\mathcal{W} \to \mathscr{T}_{y^2+y}$. ◊

Dynamical systems and graph fibrations. In Example 2.164, there's a certain relationship we can see between the graph we associate to the dynamical system $Sy^S \to p$, namely



and the trees (which are also graphs) that its mate $Sy^S \to \mathscr{T}_p$ associates to each element of S, e.g. for the green dot:



Indeed, there is a map of graphs from the latter to the former, which sends all the green dots in the tree to the green dot in the dynamical system, etc. This map of graphs is a kind of *fibration*, or maybe we should say op-fibration, in the sense that the set of arrows emanating from every dot in the tree is in bijection with the set of arrows emanating from its image in the dynamical system graph.

Exercise 2.166.

1. Draw the other two trees associated to the dynamical system in Example 2.164.
2. Do they also have an op-fibration down to the dynamical system graph?

3. Are these op-fibrations special in any way? That is, are they unique, or have any universal property? ◊

Replacing Sy^S by another comonoid. For any interface p, we defined a dependent dynamical system—also called a generalized Moore machine—to be a set S and a polynomial map $Sy^S \to p$. But now it seems that what really makes this work is that Sy^S underlies a comonoid. This suggests that we could instead have defined a dependent dynamical system to be a comonoid $\mathcal{C} = (\mathfrak{c}, \epsilon, \delta)$ together with a map $\mathfrak{c} \to p$. What are the similarities and differences?

Here are some similarities. We still get a cofunctor $F\colon \mathcal{C} \to \mathscr{T}_p$, so we associate a p-tree to each object in $\mathcal{C}$ and pass back paths out of its root to morphisms in $\mathcal{C}$. In terms of dynamics, we would think of objects in $\mathcal{C}$ as internal states. We still have the situation from (**??**), meaning that for every state $c \in \mathcal{C}$, we get a position $i \coloneqq F_1(c)$ in $p(1)$, and for every direction $d \in p[i]$ there we get a new state $\mathrm{cod}(F_c^\sharp(d)) \in \mathcal{C}$.

But in fact we get a little more from F, and this is where the differences come in. Namely, given a direction $d \in p[i]$, we get the morphism $F_c^\sharp(d)$ itself. In the state category Sy^S, there is a unique morphism between every two objects, so this passed-back morphism carries no data beyond what its codomain is. But for a more general comonoid $\mathcal{C}$, the morphisms *do* carry data.

Thus we can think of a map $\mathfrak{c} \to p$ as a dynamical system that "records its history". That is, given a path $\mathscr{T}_p$, a sequence of inputs to our dynamical system, we get a morphism in $\mathcal{C}$. If, unlike in a state category Sy^S, there are multiple morphisms between objects, we will know which one was actually taken by the system.

This seems like a nice generalization of dynamical systems—history-recording dynamical systems—and may have some use. However, we will see in Chapter 3 that there are strpmg theoretical reason to emphasize the ahistorical state categories Sy^S. For one thing, the category of all such Sy^S-style dynamical systems on p forms a topos for any $p \in \mathbf{Poly}$.

2.3.5 Some math about cofree comonoids

Proposition 2.167. For every polynomial p, the cofree category $\mathscr{T}_p$ is free on a graph. That is, there is a graph G_p whose associated free category in the usual sense (the category of vertices and paths in G_p) is isomorphic to $\mathscr{T}_p$.

Proof. For vertices, we let V_p denote the set of p-trees,

$$V_p \coloneqq \mathsf{tree}_p(1).$$

For arrows we use the map $\pi\colon \mathsf{tree}_p \to p$ from **??** to define

$$A_p \coloneqq \sum_{t \in \mathsf{tree}_p(1)} p[\pi_1(t)]$$

In other words A_p is the set $\{d \in p[\pi_1(t)] \mid t \in \mathsf{tree}_p\}$ of directions in p that emanate from the root corolla of each p-tree. The source of (t, d) is t and the target is $\mathrm{cod}(\pi_t^\sharp(d))$. It is clear that every morphism in $\mathscr{T}_t$ is the composite of a finite sequence of such morphisms, completing the proof. □

Corollary 2.168. Let p be a polynomial and $\mathscr{T}_p$ the cofree comonoid. Every morphism in $\mathcal{C}_p$ is both monic and epic.

Proof. The free category on a graph always has this property, so the result follows from Proposition 2.167. □

Proposition 2.169. The cofree functor $p \mapsto \mathscr{T}_p = (\mathsf{tree}_p, \mathrm{root}, \mathrm{focus})$ is lax monoidal; in particular there is a map of polynomials $y \to \mathsf{tree}_y$, and for any $p, q \in \mathbf{Poly}$ there is a natural map

$$\mathsf{tree}_p \otimes \mathsf{tree}_q \to \mathsf{tree}_{p \otimes q}.$$

satisfying the usual conditions.

Proof. By Proposition 2.133, the left adjoint $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is strong monoidal. A consequence of Kelly's doctrinal adjunction theorem [**kelly1974doctrinal**] says that the right adjoint of an op-lax monoidal functor is lax monoidal. □

Exercise 2.170.

1. What polynomial is tree_y?
2. What is the map $y \to \mathsf{tree}_y$ from Proposition 2.169?
3. Explain in words how to think about the map $\mathsf{tree}_p \otimes \mathsf{tree}_q \to \mathsf{tree}_{p \otimes q}$ from Proposition 2.169, for arbitrary $p, q \in \mathbf{Poly}$. ◊

Proposition 2.171. The additive monoid $y^{\mathbb{N}}$ of natural numbers has a $\times$-monoid structure in $\mathbf{Cat}^\sharp$.

Proof. The right adjoint $p \mapsto \mathscr{T}_p$ preserves products, so $y^{\mathrm{List(n)}} \cong \mathscr{T}_{y^{\mathsf{n}}}$ is the n-fold product of $y^{\mathbb{N}}$ in $\mathbf{Cat}^\sharp$. We thus want to find cofunctors $e\colon y \to y^{\mathbb{N}}$ and $m\colon y^{\mathrm{List(2)}} \to y^{\mathbb{N}}$ that satisfy the axioms of a monoid.

The unique polynomial map $y \to y^{\mathbb{N}}$ is a cofunctor (it is the mate of the identity $y \to y$). We take m to be the mate of the polynomial map $y^{\mathrm{List(2)}} \to y$ given by the list $[1, 2]$. One can check by hand that these definitions make $(y^{\mathbb{N}}, e, m)$ a monoid in $(\mathbf{Cat}^\sharp, y, \times)$. □

Proposition 2.172. The functor

$$\mathbf{Cat}^\sharp(-, y^{\mathbb{N}}) \colon \mathbf{Cat}^\sharp \to \mathbf{Mon}^{\mathrm{op}}$$

represented by $y^{\mathbb{N}}$ is right adjoint to the inclusion $\mathbf{Mon}^{\mathrm{op}} \to \mathbf{Cat}^\sharp$ from Proposition 2.90.

Proof. We saw that $y^{\mathbb{N}}$ is a monoid object in Proposition 2.171. A cofunctor $\mathcal{C} \nrightarrow y^{\mathbb{N}}$ is a policy in $\mathcal{C}$: it assigns an outgoing morphism to each object of $\mathcal{C}$. Any two such trajectories can be multiplied: we simply do one and then the other; this is the monoid operation. The policy assigning the identity to each object is the unit of the monoid. Let's denote this functor by $T \coloneqq \mathbf{Cat}^\sharp(-, y^{\mathbb{N}})$.

Let $\mathcal{C}$ be a category and $(M, e, *)$ a monoid. A cofunctor $F \colon \mathcal{C} \nrightarrow y^M$ has no data on objects; it is just a way to assign to each $c \in \mathcal{C}$ and $m \in M$ a morphism $F_c^\sharp(m) \colon c \to c'$ for some $c' \coloneqq \mathrm{cod}(F_c^\sharp(m))$. This assignment must send identities to identities and composites to composites: given $m' \in M$ we have $F_c^\sharp(m \fatsemi m') = F_c^\sharp(m) \fatsemi F_{c'}^\sharp(m')$. This is exactly the data of a monoid morphism $M \to T(\mathcal{C})$: it assigns to each $m \in M$ a policy in $\mathcal{C}$, preserving unit and multiplication. □

Proposition 2.173. There is a commutative square of left adjoints

$$\begin{array}{ccc} \mathbf{Mon}^{\mathrm{op}} & \xrightarrow{U} & \mathbf{Set}^{\mathrm{op}} \\ {\scriptstyle y^-}\downarrow & & \downarrow{\scriptstyle y^-} \\ \mathbf{Cat}^\sharp & \xrightarrow[U]{} & \mathbf{Poly} \end{array}$$

where the functors denoted U are forgetful functors.

Proof. Using the fully faithful functor $y^- \colon \mathbf{Mon}^{\mathrm{op}} \leftrightarrows \mathbf{Cat}^\sharp$ from Proposition 2.90, it is easy to check that the above diagram commutes.

The free-forgetful adjunction $\mathbf{Set} \leftrightarrows \mathbf{Mon}$ gives an opposite adjunction $\mathbf{Set}^{\mathrm{op}} \leftrightarrows \mathbf{Mon}^{\mathrm{op}}$, where U is now left adjoint. We saw that $y^- \colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{Poly}$ is a left adjoint in Proposition 1.183, that $U \colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is a left adjoint in Theorem 2.159, and that $y^- \colon \mathbf{Mon} \to \mathbf{Cat}^\sharp$ is a left adjoint in Proposition 2.172. □

Chapter 3

Data dynamics

3.1 Introduction

In the previous chapter we saw that comonoids in **Poly** are categories, but that morphisms of comonoids are not functors; they're called cofunctors. If someone were to ask "which are better, functors or cofunctors", the answer is clear. Functors are fundamental to mathematics itself, relating branches from set theory to logic to algebra to measure theory, etc. Cofunctors don't have anywhere near that sort of reach in terms of applicability. Still they provide an interesting way to compare categories, as well as new invariants of categories, like the monoid of direction fields on a category.

In this chapter we'll consider another kind of morphism between comonoids in **Poly**, i.e. categories, and one that is a bit more familiar than cofunctors. Namely, we'll consider the bimodules between comonoids. One might want to call them bi-comodules, but this name is just a bit too long and the name bimodule is not ambiguous, so we'll go with it. So why do I say they're more familiar?

It turns out that bimodules between comonoids in **Poly** (categories) are also important objects of study in category theory. If $\mathcal{C}$ and $\mathcal{D}$ are categories, Richard Garner showed that a bimodule between them can be identified with what's known as a *parametric right adjoint* between the associated copresheaf categories $\mathcal{C}$-**Set** and $\mathcal{D}$-**Set**. Parametric right adjoints, or *pra*'s come up in ∞-category theory, but they also have a much more practical usage: they are the so-called *data migration functors*.

Indeed, we'll make due on a claim we made in **??**, that there is a strong connection between the basic theory of databases and the theory of bimodules in **Poly**. Databases have two parts: they have a schema, a specification of various types and relationships between them, and an instance, which is actual data sorted into those types and having those relationships. As we'll see, one can formalize the schema as a category $\mathcal{C}$ and the instance as a functor $I\colon \mathcal{C} \to$ **Set**. Here's an example of a theorem we'll prove:

Theorem 3.1. For a comonoid $\mathscr{C} = (c, \epsilon, \delta)$, also understood as a category $\mathcal{C}$, the following categories are equivalent:

1. functors $\mathcal{C} \to$ **Set**;
2. discrete opfibrations over $\mathcal{C}$;
3. cartesian cofunctors to $\mathscr{C}$;
4. linear left $\mathscr{C}$-modules;
5. constant left $\mathscr{C}$-modules;
6. $(\mathscr{C}, 0)$-bimodules;
7. representable right $\mathscr{C}$-modules;
8. $\mathscr{C}$-coalgebras (sets with a coaction by $\mathscr{C}$).

Moreover, up to isomorphism, a $\mathscr{C}$-coalgebra can be identified with a dynamical system with comonoid interface $\mathscr{C}$.

The proof will be given in Theorem 3.28.

The plan of the chapter is as follows. We'll begin in Section 3.2 by reviewing copresheaves on a category, and their relationship to databases and dynamical systems. Then in Section 3.3 we'll prove a number of theoretical results, including Theorem 3.28 and Garner's "bimodules are parametric right adjoints" result. We'll continue to give intuition and applications in database and dynamical systems theory. Finally in Section 3.5 we'll provide some looser discussion and lay out some open questions.

3.2 Copresheaves, databases, and dynamical systems

Let $\mathcal{C}$ be a small category. One of the most important constructions in category theory is that of the category of copresheaves on $\mathcal{C}$.[1] This is the category

$$\mathcal{C}\text{-}\mathbf{Set} \coloneqq \mathbf{Fun}(\mathcal{C}, \mathbf{Set})$$

whose objects are functors $\mathcal{C} \to$ **Set** and whose morphisms are natural transformations between them.

Example 3.2. Suppose $(G, e, *)$ is a monoid (e.g. a group). In a first course on abstract algebra, one encounters the notion of a *G-set*, which is a set X together with a G action: for every element $g \in G$ we get a function $\alpha_g \colon X \to X$; we might write $\alpha_g(x)$ as $g \cdot x$. To be a G-action, the $\cdot$ operation needs to satisfy two rules: $e \cdot x = x$ and $g \cdot (h \cdot x) = (g * h) \cdot x$. A morphism between two G-sets (sets X and Y, each equipped with a G-action) is just a function $f \colon X \to Y$ that satisfies a single rule: $f(g \cdot x) = g \cdot (f(x))$ for all $g \in G$ and $x \in X$.

Now recall that any monoid (e.g. a group) G can be understood as a category with one object, let's call our category $\mathcal{G}$ and the unique object $\blacktriangle$. The elements of G,

[1] Many would say that presheaves on $\mathcal{C}$ are more fundamental, but since the notions are equivalent—just use $\mathcal{C}^{\mathrm{op}}$ to switch between them—we will consider the difference moot. We will focus on copresheaves.

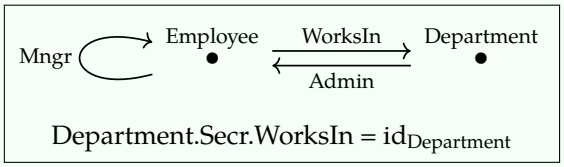
including the identity and the multiplication, are encoded as the morphisms ▲ → ▲ in $\mathcal{G}$.

It turns out that G-sets are precisely functors $F\colon \mathcal{G} \to \mathbf{Set}$: the set $F(▲)$ is our X above, and since the elements of G are now morphisms $g\colon ▲ \to ▲$, the functor F sends them to functions $F(g)\colon X \to X$; this is the $g \cdot -$ operation. The axioms of a functor—preservation of identities and compositions—ensure the two rules of the $\cdot$ operation. Finally, morphisms between G-sets are exactly the natural transformations between functors; the naturality condition becomes the rule $f(g \cdot x) = g \cdot (f(x))$ we saw above.

G-sets are copresheaves on the associated one-object category $\mathcal{G}$.

Exercise 3.3. Consider the category $\mathcal{C}$ shown here:

$\mathcal{C} :=$



It is generated by two objects, three morphisms, and the equation shown above.

1. What is its emanation polynomial?

Consider now the database instance shown here:

Employee	WorksIn	Mngr
Alice	IT	Alice
Bobby	IT	Alice
Carla	Sales	Carla

Department	Admin
IT	Bobby
Sales	Carla

Consider this database instance as a functor $S\colon \mathcal{C} \to \mathbf{Set}$.

2. Say what sets S assigns the two objects.
3. Say what functions S assigns the three generating morphisms. ◊

Definition 3.4 (Discrete opfibration)**.** Let $\mathcal{C}$ be a category. A pair $(\mathcal{S}, \pi)$, where $\mathcal{S}$ is a category and $\pi\colon \mathcal{S} \to \mathcal{C}$ is a functor, is called a *discrete opfibration over* $\mathcal{C}$ if it satisfies the following condition.

- for every object $s \in \mathcal{S}$, object $c' \in \mathcal{C}$, and morphism $f\colon \pi(s) \to c'$ there exists a unique object $s' \in \mathcal{S}$ and morphism $\bar{f}\colon s \to s'$ such that $\pi(s') = c'$ and $\pi(\bar{f}) = f$.

$$\begin{array}{ccc} s & \overset{\bar{f}}{\dashrightarrow} & s' \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle\pi} \\ \pi(s) & \xrightarrow[f]{} & c' \end{array}$$

A *morphism* $(\mathcal{S}, \pi) \to (\mathcal{S}', \pi')$ between discrete opfibrations over $\mathcal{C}$ is a functor $F\colon \mathcal{S} \to$

$\mathcal{S}'$ making the following triangle commute:

$$\begin{array}{ccc} \mathcal{S} & \xrightarrow{F} & \mathcal{S}' \\ & {\scriptstyle\pi}\searrow \quad \swarrow{\scriptstyle\pi'} & \\ & \mathcal{C} & \end{array} \tag{3.5}$$

We denote the category of discrete opfibrations into $\mathcal{C}$ by $\mathbf{dopf}(\mathcal{C})$.

Exercise 3.6. Show that if $F : \mathcal{S} \to \mathcal{S}'$ is a functor making the triangle (3.5) commute, then F is also a discrete opfibration. ◊

Exercise 3.7. Suppose $\pi : \mathcal{S} \to \mathcal{C}$ is a discrete opfibration and $i \in \mathcal{S}$ is an object. With notation as in Definition 3.4, show the following:

1. Show that the lift $\overline{\mathrm{id}}_{\pi(i)} = \mathrm{id}_i$ of the identity on $\pi(i)$ is the identity on i.
2. Show that for $f : \pi(i) \to c$ and $g : c \to c'$, we have $\bar{f} \fatsemi \bar{g} = \overline{f \fatsemi g}$.
3. Show that π is a cofunctor. ◊

Definition 3.8 (Category of elements $\int^{\mathcal{C}} I$)**.** Given a functor $I : \mathcal{C} \to \mathbf{Set}$, its category of elements $\int^{\mathcal{C}} I$ is defined to have objects

$$\mathrm{Ob}(\textstyle\int^{\mathcal{C}} I) \coloneqq \{(c, x) \mid c \in \mathrm{Ob}(\mathcal{C}), x \in I(c)\}$$

and given two objects (c, x) and (c', x'), the hom-set is given by

$$\mathrm{Hom}((c, x), (c', x')) \coloneqq \{f : c \to c' \mid I(f)(x) = x'\}.$$

Identities and composites in $(\int^{\mathcal{C}} I)$ are inherited from $\mathcal{C}$.

There is a functor $\pi : (\int^{\mathcal{C}} I) \to \mathcal{C}$ sending $\pi(c, x) \coloneqq c$ and $\pi(f) \coloneqq f$.

Exercise 3.9. Show that if $I : \mathcal{C} \to \mathbf{Set}$ is a functor then the functor $\pi : (\int^{\mathcal{C}} I) \to \mathcal{C}$ defined in Definition 3.8 is a discrete opfibration, as in Definition 3.4. ◊

Exercise 3.10. Draw the category of elements for the functor $S : \mathcal{C} \to \mathbf{Set}$ shown in Exercise 3.3. ◊

Exercise 3.11. Suppose that $I, J : \mathcal{C} \to \mathbf{Set}$ are functors and $\alpha : I \to J$ is a natural transformation.

1. Show that α induces a functor $(\int^{\mathcal{C}} I) \to (\int^{\mathcal{C}} J)$.

2. Show that it is a morphism of discrete opfibrations in the sense of Definition 3.4.
3. Show that this construction is functorial. We denote this functor by
$$\int^{\mathcal{C}} : \mathcal{C}\text{-}\mathbf{Set} \to \mathbf{dopf}(\mathcal{C})$$ ◊

Exercise 3.12. Let G be a graph, and let $\mathcal{G}$ be the free category on it. Show that for any functor $S : \mathcal{G} \to \mathbf{Set}$, the category $\int^{\mathcal{G}} S$ of elements is again free on a graph. ◊

Proposition 3.13. Let $\mathcal{C}$ be a category. The following are equivalent:
1. the category $\mathcal{C}$-**Set** of functors $\mathcal{C} \to \mathbf{Set}$,
2. the category of discrete opfibrations over $\mathcal{C}$,
3. the category of cartesian cofunctors into $\mathcal{C}$.

In fact the latter two are isomorphic.

Proof. By Exercise 3.11 we have a functor $\int^{\mathcal{C}} : \mathcal{C}\text{-}\mathbf{Set} \to \mathbf{dopf}(\mathcal{C})$. There is a functor going back: given a discrete opfibration $\pi : \mathcal{S} \to \mathcal{C}$, we define $(\partial\pi) : \mathcal{C} \to \mathbf{Set}$ on $c \in \mathrm{Ob}(\mathcal{C})$ and $f : c \to c'$ by

$$(\partial\pi)(c) \coloneqq \{s \in \mathcal{S} \mid \pi(s) = c\}$$
$$(\partial\pi)(f)(s) \coloneqq \bar{f}(s).$$

On objects, the roundtrip $\mathcal{C}$-**Set** $\to \mathcal{C}$-**Set** sends $I : \mathcal{C} \to \mathbf{Set}$ to the functor

$$\begin{aligned} c \mapsto \{s \in \textstyle\int^{\mathcal{C}} I \mid \pi(s) & \\ = \{(c, x) \mid x \in I(c)\} & \qquad = I(c). \end{aligned}$$

The roundtrip $\mathbf{dopf}(\mathcal{C}) \to \mathbf{dopf}(\mathcal{C})$ sends $\pi : \mathcal{S} \to \mathcal{C}$ to the discrete opfibration whose object set is $\{(c, s) \in \mathrm{Ob}(\mathcal{C}) \times \mathrm{Ob}(\mathcal{S}) \mid \pi(s) = c\}$ and this set is clearly in bijection with $\mathrm{Ob}(\mathcal{S})$. Proceeding similarly, one defines an isomorphism of categories $\mathcal{S} \cong \int^{\mathcal{C}} \partial\pi$.

The above correspondence is well-known; it remains to address the relationship between (2) and (3). A cartesian cofunctor $(\varphi_1, \varphi^\sharp) : \mathcal{S} \nrightarrow \mathcal{C}$ gives a function $\varphi : \mathrm{Ob}(\mathcal{S}) \to \mathrm{Ob}(\mathcal{C})$ and for each $s \in \mathcal{S}$ an isomorphism

$$\varphi^\sharp_s : \mathcal{C}[\varphi_1(s)] \xrightarrow{\cong} \mathcal{S}[s]$$

between the set of $\mathcal{S}$-morphisms emanating from s and the set of $\mathcal{C}$-morphisms emanating from $\varphi_1(s)$. This isomorphism respects identities, codomains, and composites. As such we can define a functor that acts as φ_1 on objects and $(\varphi^\sharp)^{-1}$ on morphisms, and it is easily checked to be a discrete opfibration.

Finally, given a discrete opfibration $\pi : \mathcal{S} \to \mathcal{C}$, we define $\varphi_1 \coloneqq \mathrm{Ob}(\pi)$ to be its on-objects part, and for any $s \in \mathrm{Ob}(\mathcal{S})$ and emanating morphism $f \in \mathcal{C}[\varphi_1(s)]$, we define $\varphi^\sharp_s(f) \coloneqq \bar{f}$ to be the lift guaranteed by Definition 3.4. The conversions between discrete opfibrations and cartesian cofunctors are easily seen to be functorial and the roundtrips are identities. □

Recall from **??** that a dynamical system on a category $\mathcal{C}$ consists of a set S and a cofunctor $Sy^S \nrightarrow \mathcal{C}$ from the state category on S to $\mathcal{C}$.

Proposition 3.14 (Database instances are dynamical systems)**.** Up to isomorphism, discrete opfibrations into $\mathcal{C}$ can be identified with dynamical systems on $\mathcal{C}$.

In case it isn't clear, this association is only functorial on the groupoid of objects and isomorphisms.

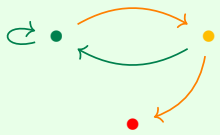
Proof. Given a discrete opfibration $\pi : \mathcal{S} \to \mathcal{C}$, take $S := \mathrm{Ob}(\mathcal{S})$ and define $(\varphi_1, \varphi^\sharp) : Sy^S \to \mathfrak{c}$ by $\varphi_1 = \pi$ and with $\varphi^\sharp$ given by the lifting: $\varphi(g) := \hat{g}$ as in Definition 3.4. One checks using Exercise 3.7 that this defines a cofunctor.

Conversely, given a cofunctor $(\varphi_1, \varphi^\sharp) : Sy^S \to \mathfrak{c}$, the function φ_1 induces a map of polynomials $Sy \to \mathfrak{c}$, and we can factor it as a vertical followed by a cartesian $Sy \to \mathfrak{s} \xrightarrow{\psi} \mathfrak{c}$. We can give $\mathfrak{s}$ the structure of a category such that ψ is a cofunctor; see Exercise 3.15. □

Exercise 3.15. With notation as in Proposition 3.14, complete the proof as follows.

1. Check that $(\varphi, \varphi^\sharp)$ defined in the first paragraph is indeed a cofunctor.
2. Find a comonoid structure on $\mathfrak{s}$ such that ψ is a cofunctor, as stated in the second paragraph.
3. Show that the two directions are inverse, up to isomorphism. ◊

Example 3.16. In Example 2.164 we had a dynamical system with $S := \{\bullet, \bullet, \bullet\}$ and $p := y^2 + y$, and $f : Sy^S \to p$ from Exercise 1.129, depicted here again for your convenience:

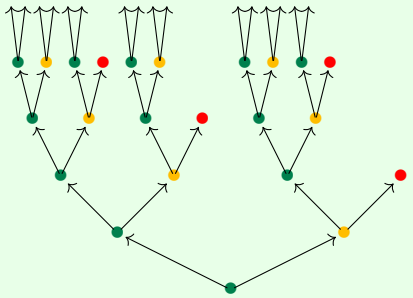


(3.17)

The polynomial map f induces a cofunctor $F : Sy^S \nrightarrow \mathscr{T}_p$ from the state category on S to the category of p-trees. We can now see this as a database instance on $\mathscr{T}_p$, considered as a database schema.

The cofree category $\mathscr{T}_t$ is actually the free category on a graph, as we saw in Proposition 2.167, and so the schema is easy. There is one table for each tree (object in

$\mathscr{T}_p$), e.g. we have a table associated to this tree:



The table has two columns, say left and right, corresponding to the two arrows emanating from the root node. The left column refers back to the same table, and the right column refers to another table (the one corresponding to the yellow dot).

Again, there are infinitely many tables in this schema. Only three of them have data in them; the rest are empty. We know in advance that this instance has three rows in total, since $|S| = 3$.

Given a dynamical system $Sy^S \to p$, we extend it to a cofunctor $\varphi\colon Sy^S \nrightarrow \mathscr{T}_p$. By Propositions 3.13 and 3.14, we can consider it as a discrete opfibration over $\mathscr{T}_p$. By Exercise 3.12 the category $\int\varphi$ is again free on a graph. It is this graph that we usually draw when depicting the dynamical system, e.g. in (3.17).

Exercise 3.18. Give an example of a dynamical system on $p \coloneqq y^2 + y$ for which the corresponding database instance has a table with at least two rows. ◊

Exercise 3.19. A dynamical system with interface y is a map $Sy^S \to y$.

1. What is the corresponding database schema?
2. Explain what the corresponding database instance looks like.
3. In particular, how many total rows does it have?
4. Give an example with $|S| = 7$, displayed both as a dynamical system (with states and transitions) and as a database instance. ◊

3.3 Bimodules

One might think that the database story—or you could call it the copresheaves story—is somewhat tacked on to the main line here: cartesian cofunctors $\mathcal{S} \nrightarrow \mathcal{C}$ can be seen as database instances on $\mathcal{C}$, but perhaps this is just an incidental curiosity. In this section we'll see that in fact copresheaves, i.e. set-valued functors on categories, are a major aspect of the story.

In a monoidal category, one can not only consider monoids and comonoids, but also modules between these. For example, in the monoidal category of abelian groups

under bilinear product, the monoid objects are rings and the bimodules are bimodules in the usual sense. Here we are interested in comonoids rather than monoids, so we should be interested in bi-comodules; we will just call these bimodules for linguistic convenience.

We will see that bimodules in **Poly** are equivalent to data-migration functors. Given comonoids/categories $\mathcal{C}$ and $\mathcal{D}$, a bimodule $\mathcal{C} \overset{m}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathcal{D}$ is a way of moving database instances on $\mathcal{D}$ to database instances on $\mathcal{C}$. One could call it a "$\mathcal{D}$-indexed union of conjunctive queries."

Definition 3.20 (Bimodule)**.** Let $\mathcal{C} = (\mathfrak{c}, \epsilon, \delta)$ and $\mathcal{D} = (\mathfrak{d}, \epsilon, \delta)$ be comonoids. A $(\mathcal{C}, \mathcal{D})$-*bimodule* (m, λ, ρ), denoted $\mathcal{C} \overset{m}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathcal{D}$ or in shorthand $\mathfrak{c} \overset{m}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathfrak{d}$,[a] consists of

1. a polynomial $m \in \mathbf{Poly}$,
2. a morphism $\mathfrak{c} \triangleleft m \overset{\lambda}{\leftarrow} m$, and
3. a morphism $m \overset{\rho}{\to} m \triangleleft \mathfrak{d}$,

satisfying the following commutative diagrams:

$$\begin{array}{ccc} \mathfrak{c} \triangleleft m & \xleftarrow{\lambda} & m \\ {\scriptstyle \epsilon \triangleleft m}\downarrow & \swarrow & \\ m & & \end{array} \qquad \begin{array}{ccc} \mathfrak{c} \triangleleft m & \xleftarrow{\lambda} & m \\ {\scriptstyle \delta \triangleleft m}\downarrow & & \downarrow{\scriptstyle \lambda} \\ \mathfrak{c} \triangleleft \mathfrak{c} \triangleleft m & \xleftarrow[\mathfrak{c} \triangleleft \lambda]{} & \mathfrak{c} \triangleleft m \end{array} \tag{3.21}$$

$$\begin{array}{ccc} m & \xrightarrow{\rho} & m \triangleleft \mathfrak{d} \\ & \searrow & \downarrow{\scriptstyle m \triangleleft \epsilon} \\ & & m \end{array} \qquad \begin{array}{ccc} m & \xrightarrow{\rho} & m \triangleleft \mathfrak{d} \\ {\scriptstyle \rho}\downarrow & & \downarrow{\scriptstyle m \triangleleft \delta} \\ m \triangleleft \mathfrak{d} & \xrightarrow[\rho \triangleleft \mathfrak{d}]{} & m \triangleleft \mathfrak{d} \triangleleft \mathfrak{d} \end{array} \tag{3.22}$$

$$\begin{array}{ccc} m & \xrightarrow{\rho} & m \triangleleft \mathfrak{d} \\ {\scriptstyle \lambda}\downarrow & & \downarrow{\scriptstyle \lambda \triangleleft \mathfrak{d}} \\ \mathfrak{c} \triangleleft m & \xrightarrow[\mathfrak{c} \triangleleft \rho]{} & \mathfrak{c} \triangleleft m \triangleleft \mathfrak{d} \end{array} \tag{3.23}$$

Just (m, λ) and the first line of diagrams is called a left $\mathcal{C}$-module, and similarly just (m, ρ) and the second line of diagrams is called a right $\mathfrak{D}$-module.

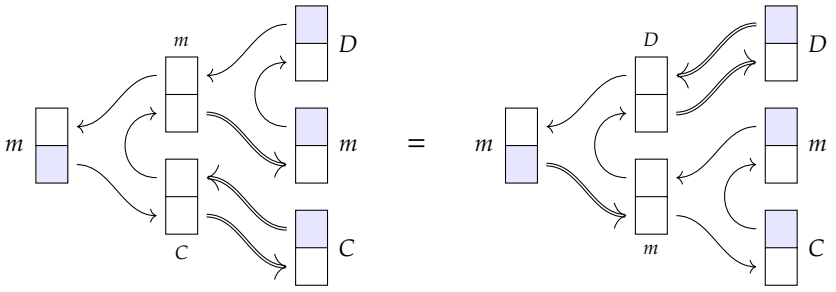
A *morphism* of $(\mathcal{C}, \mathcal{D})$-bimodules is a map $m \to n$ making the following diagram commute:

$$\begin{array}{ccccc} \mathfrak{c} \triangleleft m & \longleftarrow & m & \longrightarrow & m \triangleleft \mathfrak{d} \\ \downarrow & & \downarrow & & \downarrow \\ \mathfrak{c} \triangleleft n & \longleftarrow & n & \longrightarrow & n \triangleleft \mathfrak{d} \end{array}$$

We denote the category of $(\mathcal{C}, \mathcal{D})$-bimodules by ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$.

[a] Note that the symbol $\mathfrak{c} \triangleleft\!\!\longrightarrow\!\!\triangleleft \mathfrak{d}$ looks like it's going backwards, from $\mathfrak{d}$ to $\mathfrak{c}$; this is good because we'll see that this is the direction of data migration for a $(\mathcal{C}, \mathcal{D})$-bimodule. But the symbology is also mnemonically good because the $\triangleleft$'s go in the correct direction $\mathfrak{c} \triangleleft m \leftarrow m$ and $m \to m \triangleleft \mathfrak{d}$.

We draw the commutativity of Eq. (3.23) using polyboxes.

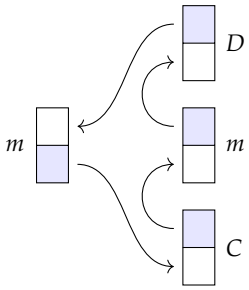


(3.24)

Exercise 3.25.

1. Draw the equations of Eq. (3.21) using polyboxes.
2. Draw the equations of Eq. (3.22) using polyboxes. ◊

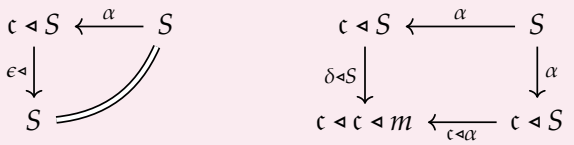
Note that Eq. (3.24) means that we can unambiguously write



Exercise 3.26. Let $\mathcal{C} = (\mathfrak{c}, \epsilon, \delta)$ be a category. Recall from **??** that y has a unique category structure.

1. Show that a left $\mathcal{C}$-module is the same thing as a $(\mathcal{C}, y)$-bimodule.
2. Show that a right $\mathcal{C}$-module is the same thing as a $(y, \mathcal{C})$-bimodule.
3. Show that every polynomial $p \in \mathbf{Poly}$ has a unique (y, y)-bimodule structure.
4. Show that there is an isomorphism of categories $\mathbf{Poly} \cong {}_y\mathbf{Mod}_y$. ◊

Definition 3.27 ($\mathscr{C}$-coalgebra)**.** Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid. A *$\mathscr{C}$-coalgebra* consists of a set S and a function $\alpha : S \to \mathfrak{c} \triangleleft S$, making the two diagrams below commute:



In other words, it is a left $\mathfrak{c}$-comodule whose carrier is constant on a set. A morphism is a function $S \to T$ commuting with α, just as for comodules; see Definition 3.20.

In order to as quickly as possible orient the reader to bimodules, we begin by proving the following.

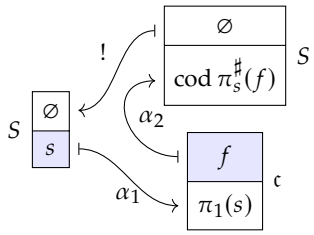
Theorem 3.28. For a comonoid $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ (category $\mathcal{C}$), the following categories are equivalent:

1. functors $\mathcal{C} \to \mathbf{Set}$;
2. discrete opfibrations over $\mathcal{C}$;
3. cartesian cofunctors to $\mathscr{C}$;
4. $\mathscr{C}$-coalgebras (sets with a coaction by $\mathscr{C}$);
5. constant left $\mathscr{C}$-modules;
6. $(\mathscr{C}, 0)$-bimodules;
7. linear left $\mathscr{C}$-modules; and
8. representable right $\mathscr{C}$-modules (opposite).

In fact, all but the first are isomorphic categories, and their core groupoid of is that of dynamical systems with interface $\mathcal{C}$.

Proof. $1 \simeq 2 \cong 3$**:** This was shown in Proposition 3.13.

$3 \cong 4$**:** Given a cartesian cofunctor $(\pi_1, \pi^\sharp) \colon \mathcal{S} \nrightarrow \mathcal{C}$, let $S := \mathrm{Ob}(\mathcal{S})$ and define a $\mathfrak{c}$-coalgebra structure $\alpha \colon S \to \mathfrak{c} \triangleleft S$ on an object $s \in \mathrm{Ob}(\mathcal{S})$ and an emanating morphism $f \colon \pi_1(s) \to c'$ in $\mathcal{C}$ by



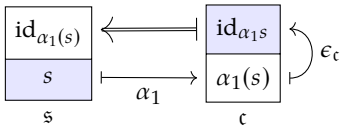
We check that this is indeed a coalgebra using properties of cofunctors. For identities in $\mathscr{C}$, we have

$$\begin{aligned}\alpha_2(s, \mathrm{id}_{\pi_1(s)}) &= \mathrm{cod}\, \pi^\sharp_s(\mathrm{id}_{\pi_1(s)}) \\ &= \mathrm{cod}\, \mathrm{id}_s = s\end{aligned}$$

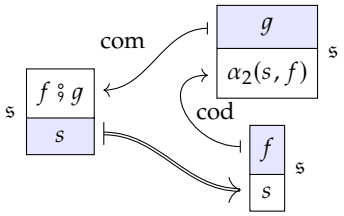
and for compositions in $\mathscr{C}$ we have

$$\begin{aligned}\alpha_2(s, f \fatsemi g) &= \mathrm{cod}\left(\pi^\sharp_s(f \fatsemi g)\right) \\ &= \mathrm{cod}\left(\pi^\sharp_s(f) \fatsemi \pi^\sharp_{\mathrm{cod}\, \pi^\sharp_s(f)} g\right) \\ &= \mathrm{cod}\left(\pi^\sharp_{\mathrm{cod}\, \pi^\sharp_s(f)} g\right) \\ &= \alpha_2(\alpha_2(s, f), g).\end{aligned}$$

Going backwards, if we're given a coalgebra $\alpha\colon S \to \mathfrak{c} \triangleleft S$, we obtain a function $\alpha_1\colon S \to \mathfrak{c} \triangleleft 1$ and we define $\mathfrak{s} := \alpha_1^* \mathfrak{c}$ and the cartesian map $\varphi := (\alpha_1, \mathrm{id})\colon \mathfrak{s} \to \mathfrak{c}$ to be the base change from Proposition 1.227. We need to show $\mathfrak{s}$ has a comonoid structure $(\mathfrak{s}, \epsilon, \delta)$ and that φ is a cofunctor. We simply define the counit $\epsilon\colon \mathfrak{s} \to y$ using α_1 and the counit on $\mathfrak{c}$:



We comultiplication $\delta\colon \mathfrak{s} \to \mathfrak{s} \triangleleft \mathfrak{s}$ using α_2 for the codomain, and using the composite $\fatsemi$ from $\mathfrak{c}$:



In Exercise 3.29 we check that $(\mathfrak{s}, \epsilon, \delta)$ really is a comonoid, that $(\alpha_1, \mathrm{id})\colon \mathfrak{s} \nrightarrow \mathfrak{c}$ is a cofunctor, that the roundtrips between cartesian cofunctors and coalgebras are identities, and that these assignments are functorial.

4 ≅ 5**:** This is straightforward and was mentioned in Definition 3.27.

5 ≅ 6**:** A right $\mathsf{0}$-module is in particular a polynomial $m \in \mathbf{Poly}$ and a map $\rho\colon m \to m \triangleleft \mathsf{0}$ such that $(m \triangleleft \epsilon) \circ \rho = \mathrm{id}_m$. This implies ρ is monic, which itself implies by Proposition 1.185 that m must be constant since $m \triangleleft \mathsf{0}$ is constant. This makes $m \triangleleft \epsilon$ the identity, at which point ρ must also be the identity. Conversely, for any set M, the corresponding constant polynomial is easily seen to make the diagrams in (3.22) commute.

5 ≅ 7**:** By the adjunction in Proposition 1.16 and the fully faithful inclusion $\mathbf{Set} \to \mathbf{Poly}$ of sets as constant polynmials, Proposition 1.173, we have isomorphisms

$$\mathbf{Poly}(Sy, \mathfrak{c} \triangleleft Sy) \cong \mathbf{Set}(S, \mathfrak{c} \triangleleft Sy \triangleleft 1) = \mathbf{Set}(S, \mathfrak{c} \triangleleft S) \cong \mathbf{Poly}(S, \mathfrak{c} \triangleleft S).$$

One checks easily that if $Sy \to \mathfrak{c} \triangleleft Sy)$ corresponds to $S \to \mathfrak{c} \triangleleft S$ under the above isomorphism, then one is a left module iff the other is.

7 ≅ 8**:** By Proposition 2.13 we have a natural isomorphism

$$\mathbf{Poly}(Sy, \mathfrak{c} \triangleleft Sy) \cong \mathbf{Poly}(y^S, y^S \triangleleft \mathfrak{c}).$$

In pictures,

The last claim was proven in Proposition 3.14. □

Exercise 3.29. Complete the proof of Theorem 3.28 ($3 \cong 4$) by proving the following.

1. Show that $(\mathfrak{s}, \epsilon, \delta)$ really is a comonoid.
2. Show that $(\alpha_1, \mathrm{id})\colon \mathfrak{s} \nrightarrow \mathfrak{c}$ is a cofunctor.
3. Show that the roundtrips between cartesian cofunctors and coalgebras are identities.
4. Show that the assignment of a $\mathscr{C}$-coalgebra to a cartesian cofunctor over $\mathcal{C}$ is functorial.
5. Show that the assignment of a cartesian cofunctor over $\mathcal{C}$ to a $\mathscr{C}$-coalgebra is functorial. ◊

Let $\mathcal{C}$ be a category. Under the above correspondence, the terminal functor $\mathcal{C} \to \mathbf{Set}$ corresponds to the identity discrete opfibration $\mathcal{C} \to \mathcal{C}$, the identity cofunctor $\mathscr{C} \to \mathscr{C}$, a certain left $\mathscr{C}$ module with carrier $\mathscr{C}(1)y$ which we call the *canonical left $\mathscr{C}$-module*, a certain constant left $\mathscr{C}$ module with carrier $\mathscr{C}(1)$ which we call the *canonical* $(\mathscr{C}, 0)$*-bimodule*, and a certain representable right $\mathscr{C}$-module with carrier $y^{\mathscr{C}(1)}$ which we call the *canonical right $\mathcal{C}$-module*.

Exercise 3.30. For any object $c \in \mathcal{C}$, consider the representable functor $\mathcal{C}(c, -)\colon \mathcal{C} \to \mathbf{Set}$. What does it correspond to as a

1. discrete opfibration over $\mathcal{C}$?
2. cartesian cofunctor to $\mathscr{C}$?
3. linear left $\mathscr{C}$-module?
4. constant left $\mathscr{C}$-module?
5. $(\mathscr{C}, 0)$-bimodule?
6. representable right $\mathscr{C}$-module?
7. dynamical system with comonoid interface $\mathscr{C}$? ◊

Exercise 3.31. We saw in Theorem 3.28 that the category ${}_{\mathcal{C}}\mathbf{Mod}_0$ of $(\mathcal{C}, 0)$-bimodule has a very nice structure: it's the topos of copresheaves on $\mathcal{C}$.

1. What is a $(0, \mathcal{C})$-bimodule?
2. What is ${}_0\mathbf{Mod}_{\mathcal{C}}$? ◊

Recall from Proposition 1.227 that for any function $f : A \to B$, we have a base-change functor $f^*\colon B\mathbf{Poly} \to A\mathbf{Poly}$ and a cartesian morphism $f^*p \to p$ for any polynomial p and isomorphism $p(1) \cong B$.

Proposition 3.32. Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid and suppose that $\rho : m \to m \triangleleft \mathfrak{c}$ is a right $\mathscr{C}$-module. Then for any set A and function $f : A \to m \triangleleft 1$, the polynomial f^*m has an induced right module structure ρ_f fitting into the commutative square below:

$$\begin{array}{ccc} f^*m & \xrightarrow{\rho_f} & f^*m \triangleleft \mathfrak{c} \\ \downarrow & & \downarrow \\ m & \xrightarrow[\rho]{} & m \triangleleft \mathfrak{c} \end{array}$$

Proof. The pullback diagram to the left defines $f^*(m)$ and that to the right is its composition with $\mathfrak{c}$

$$\begin{array}{ccc} f^*m & \longrightarrow & m \\ \downarrow & \lrcorner & \downarrow \\ A & \xrightarrow[f]{} & m \triangleleft 1 \end{array} \qquad\qquad \begin{array}{ccc} f^*m \triangleleft \mathfrak{c} & \longrightarrow & m \triangleleft \mathfrak{c} \\ \downarrow & \lrcorner & \downarrow \\ A & \xrightarrow[f]{} & m \triangleleft 1 \end{array}$$

which is again a pullback by Theorem 2.39 and Exercise 2.12. Now the map $\rho : m \to m \triangleleft \mathfrak{c}$ induces a map $\rho_f : f^*(m) \to f^*(m) \triangleleft c$; we claim it is a right module. It suffices to check that ρ_f interacts properly with ϵ and δ, which we leave to Exercise 3.33. □

Exercise 3.33. Let $\mathfrak{c}, \epsilon, \delta)$, $\rho : m \to m \triangleleft \mathfrak{c}$, and $f : A \to m \triangleleft 1$ be as in Proposition 3.32. Complete the proof of that proposition as follows:

1. Show that $\rho_f \fatsemi \epsilon = \mathrm{id}_m$
2. Show that $\rho_f \fatsemi (f^*m \triangleleft \delta) = \rho_f \fatsemi (\rho_f \triangleleft \mathfrak{c})$. ◊

Definition 3.34 (Polynomial functors of many variables)**.** GK definition.

Proposition 3.35. The bicategory of polynomial functors in the sense of **[GK]** is precisely that of bimodules between discrete categories.

Proof. ** □

Proposition 3.36. Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid in **Poly**. For any set G, the polynomial $y^G \triangleleft \mathfrak{c}$ has a natural right $\mathscr{C}$-module structure.

Proof. We use the map $(y^G \triangleleft \delta) : (y^G \triangleleft \mathfrak{c}) \to (y^G \triangleleft \mathfrak{c} \triangleleft \mathfrak{c})$. It satisfies the unitality and associativity laws because $\mathfrak{c}$ does. □

We can think of elements of G as "generators". Then if $i' : G \to \mathfrak{c} \triangleleft 1$ assigns to every generator an object of a category $\mathcal{C}$, then we should be able to find the free $\mathcal{C}$-set that i' generates.

Proposition 3.37. Functions $i' : G \to \mathfrak{c} \triangleleft 1$ are in bijection with positions $i \in y^G \triangleleft \mathfrak{c} \triangleleft 1$. Let $m := i^*(y^G \triangleleft \mathfrak{c})$ and let ρ_i be the induced right $\mathscr{C}$-module structure from Proposition 3.32. Then ρ_i corresponds to the free $\mathcal{C}$-set generated by i'.

Proof. The polynomial m has the following form:

$$m \cong y^{\sum_{g \in G} \mathfrak{c}[i'(g)]}$$

In particular ρ_i is a representable right $\mathscr{C}$-module, and we can identify it with a $\mathcal{C}$-set by Theorem 3.28. The elements of this $\mathcal{C}$-set are pairs (g, f), where $g \in G$ is a generator and $f : i'(g) \to \mathrm{cod}(f)$ is a morphism in $\mathcal{C}$ emanating from $i'(g)$. It is easy to see that the module structure induced by Proposition 3.36 is indeed the free one. □

Exercise 3.38. Let $\mathcal{C}$ be a category and $i \in \mathcal{C}$ an object.

1. Consider i as a map $y \to \mathfrak{c}$. Show that the vertical-cartesian factorization of this map is $y \to y^{\mathfrak{c}[i]} \xrightarrow{\varphi} \mathfrak{c}$.
2. Use Proposition 2.46 to show that $y^{\mathfrak{c}[i]} \triangleleft \mathfrak{c} \to \mathfrak{c} \triangleleft \mathfrak{c}$ is cartesian.
3. Show that there is a commutative square

$$\begin{array}{ccc} y^{\mathfrak{c}[i]} & \xrightarrow{\delta^i} & y^{\mathfrak{c}[i]} \triangleleft \mathfrak{c} \\ {\scriptstyle\varphi}\downarrow & \lrcorner & \downarrow{\scriptstyle\text{cart}} \\ \mathfrak{c} & \xrightarrow[\delta]{} & \mathfrak{c} \triangleleft \mathfrak{c} \end{array}$$

4. Show that this square is a pullback, as indicated.
5. Show that δ^i makes $y^{\mathfrak{c}[i]}$ a right $\mathscr{C}$-module. ◊

The map δ^i can be seen as the restriction of $\delta : \mathfrak{c} \to \mathfrak{c} \triangleleft \mathfrak{c}$ to a single starting position.

We can extend this to a functor $\mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ that sends the object i to $y^{\mathfrak{c}[i]}$. Given a morphism $f : i \to i'$ in $\mathcal{C}$, we get a function $\mathfrak{c}[i'] \to \mathfrak{c}[i]$ given by composition with f, and hence a map of polynomials $y^{\mathfrak{c}[f]} : y^{\mathfrak{c}[i]} \to y^{\mathfrak{c}[i']}$.

Exercise 3.39.

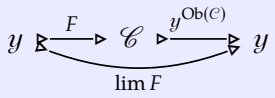
1. Show that $y^{\mathfrak{c}[f]}$ is a map of right $\mathscr{C}$-modules.
2. Show that the construction $y^{\mathfrak{c}[f]}$ is functorial in f. ◊

Definition 3.40 (Yoneda)**.** Let $\mathcal{C}$ be a category. We refer to the above functor $y^{\mathfrak{c}[-]} : \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ as the *Yoneda* functor.

Proposition 3.41. The Yoneda functor $y^{\mathfrak{c}[-]} : \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ is fully faithful.

Proof. ** □

Proposition 3.42. For any functor $F\colon \mathcal{C} \to \mathbf{Poly}$, the limit polynomial $\lim_{c\in\mathcal{C}} F(c)$ is obtained by composing with the canonical right bimodule $y^{\mathrm{Ob}(\mathcal{C})}$



Proposition 3.43. Let $\mathscr{C}$ be a comonoid. For any set I and right $\mathscr{C}$-modules $(m_i)_{i\in I}$, the coproduct $m \coloneqq \sum_{i\in I} m_i$ has a natural right-module structure. Moreover, each representable summand in the carrier m of a right $\mathscr{C}$-module is itself a right-$\mathscr{C}$ module and m is their sum.

Proof. ** □

Proposition 3.44. If $m \in \mathbf{Poly}$ is equipped with both a right $\mathscr{C}$-module and a right $\mathscr{D}$-module structure, we can naturally equip m with a $(\mathscr{C}\times\mathscr{D})$-module structure.

Proof. It suffices by Proposition 3.43 to assume that $m = y^M$ is representable. But a right $\mathscr{C}$-module with carrier y^M can be identified with a cofunctor $My^M \to \mathscr{C}$.

Thus if y^M is both a right-$\mathscr{C}$ module and a right-$\mathscr{D}$ module, then we have comonoid morphisms $\mathscr{C} \leftarrow My^M \to \mathscr{D}$. This induces a unique comonoid morphism $My^M \to (\mathscr{C}\times\mathscr{D})$ to the product, and we identify it with a right-$(\mathscr{C}\times\mathscr{D})$ module on y^M. □

3.3.1 Morphisms between bimodules

Proposition 3.45. The category ${}_{\mathfrak{d}}\mathbf{Mod}_{\mathfrak{c}}$ is a rig category: it has a terminal object (carried by $\mathfrak{d} \triangleleft 1$), an initial object (carried by 0), as well as products that distribute over coproducts.

Proof. For bimodules $\mathfrak{d} \overset{m}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{c}$ and $\mathfrak{d} \overset{n}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{c}$, the coproduct has carrier $m+n$, and the product has carrier $m \times_{d\triangleleft 1} n$. **Finish** □

Adjoint bimodules

Exercise 3.46. Recall that there is a unique $(0,\mathcal{C})$-bimodule, namely $0 \overset{0}{\triangleleft\!\!-\!\!\triangleleft} \mathcal{C}$.

1. Show that 0 has a left adjoint $\mathcal{C} \triangleleft\!\!-\!\!\triangleleft 0$; what is its carrier?
2. Show that 0 has a right adjoint $\mathcal{C} \triangleleft\!\!-\!\!\triangleleft 0$; what is its carrier? ◊

3.3.2 Bimodules as data migration functors

Proposition 3.47. Let $\mathcal{C}$ and $\mathcal{D}$ be categories; the following conditions on a functor $F\colon \mathcal{C}$-**Set** $\to \mathcal{D}$-**Set** are equivalent.

1. F is composition with a $(\mathcal{D},\mathcal{C})$-bimodule.
2. F is a parametric right adjoint in the sense of [].
3. F ... $\Sigma\Pi\Delta$
4. F profunctor + discrete opfibration
5. F preserves connected limits.

Definition 3.48. Pra-functors Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *pra-functor* (also called a *parametric right adjoint functor*) $\mathcal{C}$-**Set** $\to \mathcal{D}$-**Set** is one satisfying any of the conditions of Proposition 3.47

When a polynomial

$$m \coloneqq \sum_{i \in m(1)} y^{m[i]}$$

is given the structure of a $(\mathcal{D},\mathcal{C})$-bimodule, the symbols in that formula are given a hidden special meaning:

$$m(1) \in \mathcal{D}\text{-}\mathbf{Set} \qquad \text{and} \qquad m[i] \in \mathcal{C}\text{-}\mathbf{Set}$$

Thus $m(1)$ is a database instance on $\mathcal{D}$; in particular, each position in $i \in m(1)$ is a row in that instance. And each $m[i]$ is a database instance on $\mathcal{C}$; in particular, each direction $d \in m[i]$ is a row in that instance.

Before we knew about bimodule structures, what we called positions and directions—and what we often think of as outputs and inputs of a system—were understood as each forming an ordinary set. In the presence of a bimodule structure, the positions $m(1)$ have been organized into a $\mathcal{D}$-set and the directions $m[i]$ have been organized into a $\mathcal{C}$-set for each position i. We are listening for $\mathcal{C}$-sets and positioning ourselves in a $\mathcal{D}$-set.

Theorem 3.49. The 2-functor Copsh: **Mod** $\to$ **Cat** given by $\mathcal{C} \mapsto {}_{\mathcal{C}}\mathbf{Mod}_0$ is fully faithful.

Proof. The functor Copsh sends $\mathcal{C}$ to the corresponding copresheaf category $\mathcal{C}$-**Set** by Theorem 3.28.

Finish □

Exercise 3.50. Consider the query on $y^{\mathbb{N}}$ from **??** asking for siblings.

1. Write out the corresponding $y^{\mathbb{N}}$ instance in table form.
2. Draw it as a dynamical system.

◊

3.3.3 Monoidal operations

Theorem 3.51. For any categories $\mathcal{C}$ and $\mathcal{D}$, the functor

$$ {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \xrightarrow{\cong} \mathbf{Cat}(\mathcal{C},\ {}_{y}\mathbf{Mod}_{\mathcal{D}}) $$

is an equivalence.

Proof. □

Corollary 3.52. For any categories $\mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{D}$, the functor

$$ {}_{\mathcal{C}_1 \otimes \mathcal{C}_2}\mathbf{Mod}_{\mathcal{D}} \to \mathbf{Cat}(\mathcal{C}_1,\ {}_{\mathcal{C}_2}\mathbf{Mod}_{\mathcal{D}}) $$

is an equivalence.

Proof. ** $\mathcal{C}_1 \otimes \mathcal{C}_2$ is the usual product of categories ** □

Corollary 3.53. Let $\mathscr{C}$ be a comonoid. The category of left $\mathscr{C}$ modules is equivalent to the category of functors $\mathcal{C} \to \mathbf{Poly}$.

Proof. Use Theorem 3.51 with $\mathcal{D} = y$ and the equivalence ${}_{y}\mathbf{Mod}_{y} \cong \mathbf{Poly}$. □

Proposition 3.54. The monoidal operation + is a coproduct in **Mod**. That is, for any categories $\mathcal{C}, \mathcal{D}, \mathcal{E}$ there is an equivalence of categories

$$ {}_{\mathcal{C}+\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} \cong {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{E}} \times {}_{\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} $$

Proof. This follows from Corollary 3.52:

$$ \begin{aligned} {}_{\mathcal{C}+\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} &\cong \mathbf{Cat}(\mathcal{C}+\mathcal{D},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \\ &\cong \mathbf{Cat}(\mathcal{C},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \times \mathbf{Cat}(\mathcal{D},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \\ &\cong {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{E}} \times {}_{\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} \square \end{aligned} $$

In fact, + is almost a biproduct in **Mod**, except the required equivalence is replaced by an adjunction.

Proposition 3.55. For any categories $\mathcal{C}, \mathcal{D}, \mathcal{E}$ there is an adjunction

$$ {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}+\mathcal{E}} \underset{\Leftarrow}{\overset{\Rightarrow}{\longrightarrow}} {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{E}} $$

Proof. By Corollary 3.52, it suffices to show that there is an adjunction

$$ {}_y\mathbf{Mod}_{\mathcal{D}+\mathcal{E}} \underset{\longleftarrow}{\overset{\longrightarrow}{\Rightarrow}} {}_y\mathbf{Mod}_{\mathcal{D}} \times {}_y\mathbf{Mod}_{\mathcal{E}} \ . $$

** □

Proposition 3.56. If $\mathfrak{c} \overset{m_1}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}$ and $\mathfrak{c} \overset{m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}$ are bimodules, then so are

$$ \mathfrak{c} \overset{m_1+m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d} \qquad \text{and} \qquad \mathfrak{c} \overset{m_1 \times_{\mathfrak{c}(1)} m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d} $$

Proposition 3.57. If $\mathfrak{c}_1 \overset{m_1}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}_1$ and $\mathfrak{c}_2 \overset{m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}_2$ are bimodules, then so are

$$ \mathfrak{c}_1 + \mathfrak{c}_2 \overset{m_1+m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}_1 + \mathfrak{d}_2 \qquad \text{and} \qquad \mathfrak{c}_1 \otimes \mathfrak{c}_2 \overset{m_1 \otimes m_2}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d}_1 \otimes \mathfrak{d}_2 $$

3.3.4 Composing bimodules

We denote the composite of $\mathfrak{e} \overset{n}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{d} \overset{m}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{c}$ by

$$ \mathfrak{e} \overset{n \triangleleft_{\mathfrak{d}} m}{\triangleleft\!\!-\!\!\triangleleft} \mathfrak{c}. $$

Recall the Yoneda functor $y^{\mathfrak{c}[-]} \colon \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ from Definition 3.40. Using bimodule composition, we obtain the composite functor

$$ \mathcal{C} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \xrightarrow{y^{\mathfrak{c}[-]} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}} {}_y\mathbf{Mod}_{\mathcal{C}} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \xrightarrow{[} \triangleleft_{\mathcal{C}}] \ {}_y\mathbf{Mod}_{\mathcal{D}}. $$

By the cartesian closure of **Cat**, this can be identified with a functor ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \to \mathbf{Cat}(\mathcal{C}, {}_y\mathbf{Mod}_{\mathcal{D}})$.

Definition 3.58. For any category $\mathcal{C}$, define the category of *polynomials in* $\mathcal{C}$, denoted $\mathbf{Set}[\mathcal{C}]$, by

$$ \mathbf{Set}[\mathcal{C}] := {}_y\mathbf{Mod}_{\mathcal{C}} $$

Example 3.59. When $\mathcal{C} = y$, we have $\mathbf{Set}[y] \cong \mathbf{Poly}$, and for any set I considered as a discrete category Iy, we have $\mathbf{Set}[Iy]$ is the category of polynomial functors in I many variables.

For arbitrary $\mathcal{C}$, we can think of $\mathbf{Set}[\mathcal{C}]$ as a polynomial rig with variables in $\operatorname{Ob}\mathcal{C}$, but with arbitrary limits replacing the mere products you would find when $\mathcal{C}$ is a discrete category.

Proposition 3.60. For any category $\mathcal{C}$, the category $\mathbf{Set}[\mathcal{C}]$ is the free coproduct completion of $(\mathcal{C}\text{-}\mathbf{Set})^{\mathrm{op}}$.

Proof. ** □

Proposition 3.61 (Carrier functor)**.** For any category $\mathcal{C}$, the carrier functor

$$\mathbf{Set}[\mathcal{C}] \to \mathbf{Poly}$$

sending $m \to m \triangleleft \mathfrak{c}$ to $m \in \mathbf{Poly} \cong \mathbf{Set}[y]$ is given by composition with the principle $\mathcal{C}$-module $\mathcal{C} \triangleleft\xleftarrow{\check{\mathfrak{c}}}\triangleleft\ y$.

Proof. ** □

Proposition 3.62. The functor $\eta_{\mathcal{C}} \colon \mathcal{C} \to \mathbf{Set}[\mathcal{C}]$ corresponding to the unit bimodule $\mathfrak{c} \triangleleft\xleftarrow{\mathfrak{c}}\triangleleft\ \mathfrak{c}$ sends each object $i \in \mathfrak{c}(1)$ to the bimodule $y \triangleleft\xleftarrow{y^{\mathfrak{c}[i]}}\triangleleft\ \mathfrak{c}$.

Proposition 3.63. The functor $(\mathcal{C}\text{-}\mathbf{Set})^{\mathrm{op}} \to \mathbf{Set}[\mathcal{C}]$ has a left adjoint,

$$(\mathcal{C}\text{-}\mathbf{Set})^{\mathrm{op}} \underset{\Gamma}{\overset{y^-}{\underset{\Leftarrow}{\longrightarrow}}} \mathbf{Set}[\mathcal{C}]\ .$$

Proof. ** □

Theorem 3.64. For any categories $\mathcal{C}, \mathcal{D}$, there is an adjunction

$$({}_{\mathcal{C}^{\mathrm{op}} \times \mathcal{D}}\mathbf{Mod}_0)^{\mathrm{op}} \underset{\Rightarrow}{\longrightarrow} {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$$

such that the left adjoint is a fully faithful inclusion of profunctors into bimodules.

Proposition 3.65. For any category $\mathcal{C}$, the carrier functor creates limits.

Proof. ** □

Theorem 3.66. For any $(\mathcal{C}, \mathcal{D})$-bimodule m, the functor $\mathbf{Set}[\mathcal{C}] \to \mathbf{Set}[\mathcal{D}]$ defined by composition with m preserves constants, coproducts, and all small limits; in particular, it's a map of rig categories.

Proof. ** □

Proposition 3.67. The composite of a linear left $\mathcal{C}$-module and a representable right $\mathcal{C}$-module is the set of natural transformations between the corresponding copresheaves.

Proposition 3.68. For any categories $\mathcal{C}$ and $\mathcal{D}$ and $(\mathcal{D}, \mathcal{C})$-bimodule $\mathfrak{d} \xleftarrow{m} \mathfrak{c}$, we have

$$m \triangleleft_{\mathfrak{c}} \mathfrak{c} \cong m$$

where $\mathfrak{c} \xleftarrow{\mathfrak{c}} y$ is the regular left $\mathfrak{c}$-module.

Proposition 3.69. Every bimodule $\mathfrak{c} \xleftarrow{S} 0$, is isomorphic to the composite

$$\mathfrak{c} \xleftarrow{Sy} y \xleftarrow{1} 0$$

for a bimodule $\mathfrak{c} \xleftarrow{Sy} y$ with carrier $Sy \in \mathbf{Poly}$.

Moreover, Sy is adjoint to a bimodule $y \xleftarrow{y^S} \mathfrak{c}$ with carrier $y^S \in \mathbf{Poly}$.

3.4 The proarrow equipment

A type of categorical structure called pro arrow equipment gives a nice way to organize what's going on with comonoids, cofunctors, bimodules, and maps between them. A pro arrow equipment is the type of double category, so it has objects, vertical morphisms, horizontal morphisms, and 2-cells; for us, the objects will be categories, the vertical morphisms will be cofunctors, and the horizontal morphisms will be parametric right adjoints.

3.4.1 Adjoint bimodules

Proposition 3.70. A functor $F \colon \mathcal{C} \to \mathcal{D}$ gives rise to a right adjoint bimodule (= left adjoint pra) $\mathfrak{c} \xleftarrow{\Delta_F} \mathfrak{d}$.

Proof. ** □

Proposition 3.71. A cofunctor $\varphi \colon \mathcal{C} \nrightarrow \mathcal{D}$ gives rise to a left adjoint bimodule (= right adjoint pra) $\mathfrak{c} \xleftarrow{\widehat{\varphi}} \mathfrak{d}$.

Proof. ** □

Proposition 3.72. For any category $\mathcal{C}$, the pras associated to the terminal cofunctor $\mathcal{C} \nrightarrow y$ are the principal left and right $\mathscr{C}$-modules.

Example 3.73 (Types on database tables). For a category $\mathcal{C}$, a functor $\mathcal{C} \to \mathbf{Set}$ doesn't appear to have actual attributes, e.g. string, integers, etc., attached to its elements. To correct this, let's add a type T_c to each table in c, and ask that each row in c is assigned an element of T_c.

To do this, we give a functor $T\colon \mathrm{Ob}(\mathcal{C}) \to \mathbf{Set}$, i.e. $\mathrm{Ob}(\mathcal{C}) \triangleleft\!\!\xleftarrow{T}\!\!\triangleleft\ \mathsf{0}$. We also have the canonical cofunctor $o\colon \mathcal{C} \nrightarrow \mathrm{Ob}(\mathcal{C})$. Now given an arbitrary instance $\mathcal{C} \triangleleft\!\!\xleftarrow{I}\!\!\triangleleft\ \mathsf{0}$, there are two things we could do. We could push it forward along the left adjoint prafunctor $\mathrm{Ob}(\mathcal{C}) \triangleleft\!\!-\!\!\triangleleft\ \mathcal{C}$ and then map it to T.

Perhaps better would be to compose T with the right adjoint prafunctor $\mathcal{C} \triangleleft\!\!-\!\!\triangleleft\ \mathrm{Ob}(\mathcal{C}) \triangleleft\!\!\xleftarrow{T}\!\!\triangleleft\ \mathsf{0}$ and map I into that composite. The reason it is better is that the category of coalgebras I equipped with a map into a fixed coalgebra X (e.g. $X = \check{o} \triangleleft_{\mathrm{Ob}\,\mathcal{C}} T$) is equivalent to the category of coalgebras (instances) on the category of elements $\int^{\mathcal{C}} X$. So we have found that instances, even equipped with types, can be understood just in terms of $\mathcal{C}$-sets, beefing up $\mathcal{C}$ if necessary.

Proposition 3.74. A pra $\mathcal{C}\mathbf{Set} \to \mathcal{D}\mathbf{Set}$ is a left adjoint if it is of the form $\Sigma_G \circ \Delta_F$ for functors $\mathcal{C} \xleftarrow{F} \mathcal{X} \xrightarrow{G} \mathcal{D}$, where G is a discrete opfibration.

Proof. ** □

Corollary 3.75. If $\varphi\colon \mathcal{C} \nrightarrow \mathcal{D}$ is a cartesian cofunctor corresponding to a discrete opfibration $F\colon \mathcal{C} \to \mathcal{D}$, then there is an isomorphism $\widehat{\varphi} \cong \Delta_F$. In particular, this gives rise to an adjoint triple.

Proof. ** □

3.5 Discussion and open questions

In this section, we lay out some questions that whose answers may or may not be known, but which were not known to us at the time of writing. They vary from concrete to open-ended, they are not organized in any particular way, and are in no sense complete. Still we hope they may be useful to some readers.

1. What can you say about comonoids in the category of all functors $\mathbf{Set} \to \mathbf{Set}$, e.g. ones that aren't polynomial.
2. What can you say about the internal logic for the topos $\mathscr{T}_p$-$\mathbf{Set}$ of dynamical systems with interface p, in terms of p?

3. How does the logic of the topos $\mathscr{T}_p$ help us talk about issues that might be useful in studying dynamical systems?
4. Morphisms $p \to q$ in **Poly** give rise to left adjoints $\mathscr{T}_p \to \mathscr{T}_q$ that preserve connected limits. These are not geometric morphisms in general; in some sense they are worse and in some sense they are better. They are worse in that they do not preserve the terminal object, but they are better in that they preserve every connected limit not just finite ones. How do these left adjoints translate statements from the internal language of p to that of q?
5. Consider the $\times$-monoids and $\otimes$-monoids in three categories: **Poly**, $\mathbf{Cat}^\sharp$, and **Mod**. Find examples of these comonoids, and perhaps characterize them or create a theory of them.
6. Is there a functor **Poly** has pullbacks, so one can consider the bicategory of spans in **Poly**. Is there a functor from that to **Mod** that sends $p \mapsto \mathscr{T}_p$?
7. Databases are static things, whereas dynamical systems are dynamic; yet we see them both in terms of **Poly**. How do they interact? Can a dynamical system read from or write to a database in any sense?
8. Can we do database aggregation in a nice dynamic way?
9. In the theory of polynomial functors, sums of representable functors **Set** $\to$ **Set**, what happens if we replace sets with homotopy types: how much goes through? Is anything improved?
10. Are there any functors **Set** $\to$ **Set** that aren't polynomial, but which admit a comonoid structure with respect to composition $(y, \triangleleft)$?
11. Characterize the monads in poly? They're generalizations of one-object operads (which are the Cartesian ones), but how can we think about them?
12. Both functors and cofunctors give left adjoint bimodules: for functors $F\colon \mathcal{D} \to \mathcal{C}$ we use the pullback Δ_F and for cofunctors $G\colon \mathcal{C} \nrightarrow \mathcal{D}$ we use the companion as in **??**. Can we characterize left adjoint bimodules in general?
13. What limits exist in $\mathbf{Cat}^\sharp$? Describe them combinatorially.
14. Since the forgetful functor $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is faithful, it reflects monomorphisms: if $f\colon \mathcal{C} \nrightarrow \mathcal{D}$ is a cofunctor whose underlying map on emanation polynomials is monic, then it is monic. Are all monomorphisms in $\mathbf{Cat}^\sharp$ of this form?
15. At first blush it appears that **Poly** may be suitable as the semantics of a language for protocols. Develop such a language or showcase the limitations that make it impossible or inconvenient.
16. Are there polynomials p such that one use something like Gödel numbers to encode logical propositions from the topos tree_p-**Set** into a "language" that p-dynamical systems can "work with"?

Appendix D