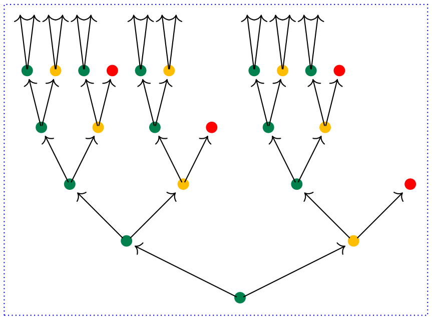
Polynomial Functors:
A General Theory of Interaction



David I. Spivak   Nelson Niu

(Last updated: July 19, 2021)

This page intentionally left blank.

David I. Spivak
Topos Institute
Berkeley, CA

Nelson Niu
University of Washington
Seattle, WA

To André Joyal

Contents

Preface

> The proposal is also intended to [serve] equally as a foundation for the academic, intellectual, and technological, on the one hand, and for the curious, the moral, the erotic, the political, the artistic, and the sheerly obstreperous, on the other.
>
> –Brian Cantwell Smith
> *On the Origin of Objects*

> For me, though, it is difficult to resist the idea that space-time is not essentially different from matter, which we understand more deeply. If so, it will consist of vast numbers of identical units—"particles of space"—each in contact with a few neighbors, exchanging messages, joining and breaking apart, giving birth and passing away.
>
> –Frank Wilczek
> *Fundamentals*

Why this book

Brief history of book

DJM visited DS in Boston for almost all of 2020. Mutual excitement over different but related things. DS super excited about **Poly**, DJM super excited about doctrines and later paradigms.

DJM encodes something like Poly in Idris and teaches Idris to DS, who having just learned Haskell (thanks Bartosz!) quickly ports the ideas to the strict **Poly** setting. We discussed writing a book together—a T-shape, consisting of a broad generalization with paradigms and a deep dive into **Poly**—after ACT2020, in July.

DS realized comonoids were important to the story. Joachim reminded DS that Richard Garner talked about cofree comonoids at a recent CT. Richard's HottEST talk was completely mind blowing for DS. Worlds collided.

Nelson joins David as undergraduate student researcher and very quickly taught David lots of cool stuff about comonoids and bimodules. Great synergy there.

DS invented poly boxes, and everything started going faster—very fast-paced and relevant research while writing a book is not so good, so it got disjointed. The writing was going quickly, if not particularly coherent, but began to falter in mid-August when DS's hands began to hurt and DJM's teaching duties started. The pace crawled.

DS personally hired Nelson as a typist. Nelson had been so helpful earlier, DS asks DJM if NN can be an author. DJM and DS realize the book would be better as two, DS peels off his part.

DS asked NN if he was willing to join in the effort to make a quality product, and he was. They started in Jan 2021. They decided to teach a course about it at the Topos Institute in Jul 2021.

Acknowledgments

Special thanks to David Jaz Myers, a brilliant colleague, a wonderfully fun conversation partner, and an all-around good guy.

Thanks go to John Baez, Spencer Breiner, Valeria de Paiva, Brendan Fong, Richard Garner, Neil Ghani, Tim Hosgood, Owen Lynch, Joachim Kock, Sophie Libkind, Dominic Orchard, Evan Patterson, Juliet Szatko, Todd Trimble, and Christian Williams.

Part I

The category of polynomial functors

Chapter 1

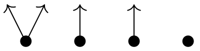
Introduction: Perspectives on polynomials

> It is a treasury box!
> Full of unexpected connections!
> It is fascinating!
> I will think about it.
>
> –André Joyal, Summer 2020,
> personal communication.

In this book we will investigate a remarkable category called **Poly**. We will see its intimate relationships with dynamic processes, decision-making, and the storage and transformation of data. But our story begins with something quite humble: high school algebra.

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

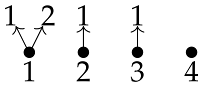
All our polynomials will involve one variable, y, chosen for reasons we'll explain soon. Polynomials in one variable can be drawn as a forest of mini-trees:



forest (1.2)

More technically, each mini-tree—a rooted tree whose *leaves* (the arrows) are all children of the *root* (the solid dot)—is called a *corolla*. So our *forests* are always unions of corollas. Each corolla in (1.2) corresponds to a *pure-power summand* of the form y^A in the polynomial given in (1.1): the corolla with 2 leaves corresponds to y^2; the two corollas with 1 leaf each correspond to the two copies of $y = y^1$; and the corolla with no leaves corresponds to $1 = y^0$.

We can label the roots and leaves of our forest, like so:



This suggests yet another depiction of the polynomial (1.1): as a *dependent set*[1] $(p[i])_{i \in I}$. Here $I = \mathsf{4} = \{1, 2, 3, 4\}$[2], the set of roots, and

$$p[1] = \mathsf{2} = \{1, 2\}, \quad p[2] = \mathsf{1} = \{1\}, \quad p[3] = \mathsf{1} = \{1\}, \quad p[4] = \mathsf{0} = \varnothing, \qquad \textit{arena} \tag{1.3}$$

so that for each root $i \in I$, the set $p[i]$ consists of the leaves at that root. We call the entire dependent set $(p[i])_{i \in I}$ an *arena*. Each element $i \in I$ is a *position* in the arena, and each element $d \in p[i]$ is a *direction* at position i. So the relationship between the arena perspective and the forest picture is that positions are roots and directions are leaves.

We will sometimes refer to the index set I as the *position-set* of p, each element $i \in I$ as a *p-position*, each set $p[i]$ as the *direction-set* at i of p, and each element $d \in p[i]$ as a *$p[i]$-direction*.

Throughout this book, we will see several other ways of interpreting polynomials. Here is a table of terminology, capturing five different perspectives from which we may view our objects of study. The first row shows the algebraic notation, as in (1.1); the second row shows the dependent set terminology, as in (1.3); the third shows the pictorial terminology of trees, as in (1.2); the fourth shows dynamical systems terminology, which we will explore in Chapter 3; and the fifth row shows decision-making terminology, which we will introduce in Section 1.2.

Polynomial Terminology

algebra	$p \coloneqq \sum_{i \in p(1)} y^{p[i]}$	$i \in p(1)$	$d \in p[i]$
dependent sets	arena	position	direction
tree pictures	corolla forest	root •	leaf ↑
dynamics	(mode-dependent) interface	output	input
decisions	menu	decision	option

(1.4)

Remark 1.5. Though a polynomial will turn out to be a *functor*, while an arena is a *dependent set*, they are so closely related that we often do not make a distinction between a polynomial p and its arena $(p[i])_{i \in I}$; they are essentially two different syntaxes for the same object. For example, we may often directly refer to the positions and directions of a polynomial, when we mean the positions and directions of its associated arena.

Exercise 1.6 (Solution here). Consider the polynomial $p \coloneqq 2y^3 + 2y + 1$ and the associated corolla forest and arena.

1. Draw the polynomial p as a corolla forest.
2. How many roots does this forest have?
3. How many positions in the arena does this represent?

---

[1] Perhaps better known as an *indexed collection* (or *family*) *of sets*. But we refer to these as *dependent sets* to compare them to sets, the same way dependent types compare to types.

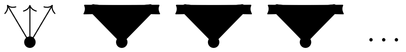
[2] In standard font, 4 represents the usual natural number. In sans serif font, $\mathsf{4}$ represents the set $\mathsf{4} = \{1, 2, 3, 4\}$ with 4 elements.

4. For each corolla in the forest, say how many leaves it has.
5. For each position in the arena, how many directions does it have? ◊

Exercise 1.7 (Solution here). Consider the polynomial $q \coloneqq y^8 + 4y$.

1. Does the polynomial q have a pure-power summand y^2?
2. Does the polynomial q have a pure-power summand y?
3. Does the polynomial q have a pure-power summand $4y$? ◊

One feature that sets our polynomials apart from the polynomials we are familiar with from high school algebra is that the coefficients and exponents are not, strictly speaking, numbers; rather, they are sets, like $1 = \{1\}$ and $2 = \{1, 2\}$. In fact, they can be arbitrary sets, as in $By^A + Dy^C$ for sets A, B, C, D, including infinite ones, as in $\mathbb{R}y^{\mathbb{N}} + 2^{\mathbb{R}}y^{\mathbb{R}}$. Any finite or infinite sum of pure-power summands, each with a finite or infinite set as an exponent, is still a polynomial. Of course, this makes their forests rather unwieldy to draw, but they can be approximated. We sketch the polynomial $y^3 + \mathbb{N}y^{[0,1]}$ as a forest below.



Exercise 1.8 (Solution here). If you were a suitor choosing the corolla forest 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. $(\mathbb{N}y)^{\mathbb{N}}$
6. Sy^S
7. $y^{100} + y^2 + 3y$
8. $y + 2y^4 + 3y^9 + 4y^{16} + \cdots$
9. Your polynomial's name p here.

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

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, decisions, and data. We speak superlatively of **Poly**:

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

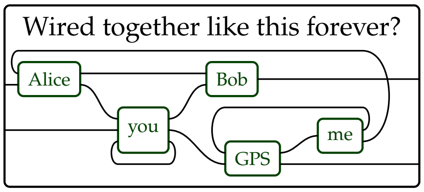
but we have not yet begun to deliver. So let's introduce some of the applications and mathematics to come.

1.1 Dynamical systems

You may already be familiar with 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 this book, we'll be looking mainly at deterministic systems, but with a lot of interesting new options:

1. The interface of the system—the way in which it can be interacted with—can change shape through time.
2. The wiring diagram connecting a bunch of systems can change through 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.



(1.9)

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 Wi-Fi 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 1.10. 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 wiring 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 3.63 to 3.65).

Exercise 1.11 (Solution here). 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 setting. ◊

But there's more that's intuitively wrong or limiting about the picture in (1.9). 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. Our ports themselves change in time. 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 1.12. 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 $y^{\mathsf{n}}+y$, which intuitively consists of two settings: one with n-many options, and the other with only one option. For simplicity, we could assume $n = 2$, so that each eyeball makes a single yes-no distinction whenever it's open.

The point, however, is that any other eyeball may be capable of noticing 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. In Section 1.2 we'll show how it relates to the making of decisions. First a quick remark.

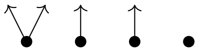
Remark 1.13. We ended Example 1.12 by joking about "real-world applications," because a pond of eyeballs is about the most bizarre thing one can imagine. But recall Nobel physicist Frank Wilczek's quote from the preface:

> For me, though, it is difficult to resist the idea that space-time is not essentially different from matter, which we understand more deeply. If so, it will consist of vast numbers of identical units—"particles of space"—each in contact with a few neighbors, exchanging messages, joining and breaking apart, giving birth and passing away.

Suppose the world was made out of a vast number of identical units, each with its own behavior, able to connect and disconnect with neighbors, and even disappear from the world of cause and effect. We may not even be interested in what our world is actually made of—just what these units are able to do. Is there such an elementary unit that could produce all other dynamical systems? The $y^2 + y$ eyeballs give a sense of a very simple interface—open and perceiving a single distinction about the world, or closed and making no distinctions—that we could imagine building an entire world from.

1.2 Decisions

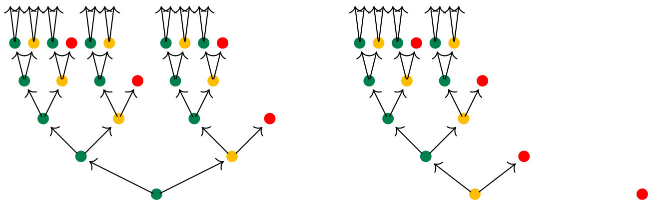
We return to the example polynomial $y^2 + 2y + 1$ from (1.1) and its corresponding corolla forest, in which positions are expressed as roots and directions are represented as leaves:



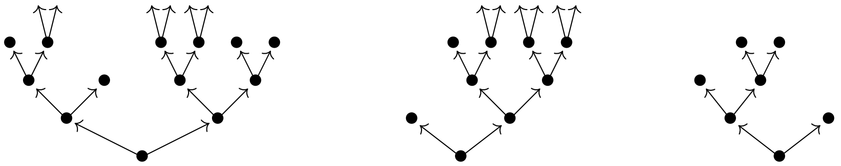
(1.14)

Concretely, we might think of each position as representing a *decision*. Associated to every decision is a set of *options* (directions). The three decisions we exhibit in (1.14) are particularly interesting: they respectively have two options, one option, 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: an impossible decision, a sort of "dead end." While the corollas $1, y$, and y^2 are each interesting as decisions, the sum $y^2 + 2y + 1$ has very little theoretical interest; it's just an example.

Now consider the following three trees, the first two of which 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 level-3 abbreviation of a finite stream):

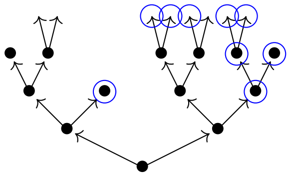


The set of such decision streams forms the objects of a category with very nice properties (it's a topos), which we call $\mathbf{Sys}(y^2+1)$. The idea is that every corolla in these diagrams has either two options, corresponding to y^2, or no options, corresponding to $1 \cong y^0$. We say that each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

Exercise 1.15 (Solution here).

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 an induced subgraph of the decision stream with the property that if n is a 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 type p decision stream.

1.3 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

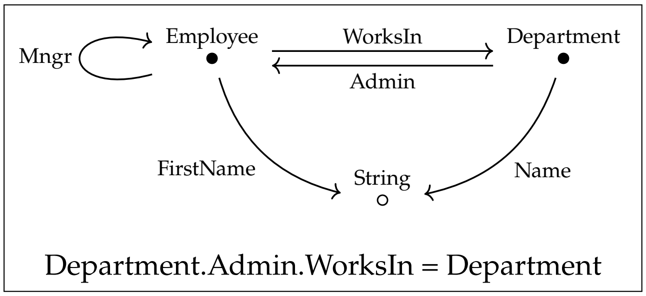
Employee	**FirstName**	**WorksIn**	**Mngr**
1	Alan	101	2
2	Ruth	101	2
3	Carla	102	3

Department	**Name**	**Admin**
101	Sales	1
102	IT	3

(1.16)

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

$\mathcal{C} :=$



(1.17)

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 (1.17) is a *finitely presented category*. The objects of the category are the points, while each arrow corresponds to a morphism of $\mathcal{C}$ (there are additional morphisms, including identity morphisms, that are not depicted). The equation at the bottom indicates that composing the morphism Admin with the morphism WorksIn yields the identity morphism on the object Department.

The database instance presented in (1.16) then corresponds to a functor $I \colon \mathcal{C} \to \mathbf{Set}$. For example, I sends the object Employee $\in \mathcal{C}$ to the set $I(\text{Employee}) := \{1, 2, 3\}$, the entries in the Employee column of the left table.

The functor also sends the morphism $\text{Mngr} \colon \text{Employee} \to \text{Employee}$ to the function $I(\text{Mngr}) \colon \{1, 2, 3\} \to \{1, 2, 3\}$ that sends each entry in the Employee column to its corresponding entry in the Mngr column. So $I(\text{Mngr})(1) = 2$, $I(\text{Mngr})(2) = 2$, and $I(\text{Mngr})(3) = 3$.

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.

Exercise 1.18 (Solution here). As above, we define the finitely presented category $\mathcal{C}$ according to (1.17) and the copresheaf I on $\mathcal{C}$ according to (1.16).

1. What is $I(\text{Department})$?
2. What is $I(\text{Admin})$?
3. Composing Admin with FirstName yields a morphism from Department to String that we denote by Admin.FirstName. What is $I(\text{Admin.FirstName})$?
4. Say we require that managers work in the same department as the employees they oversee. Write down an equation in $\mathcal{C}$ (like the one at the bottom of (1.17)) that expresses this condition.
5. How might we define $I(\text{String})$? ◊

There's a very important thing that we do with databases: we query them. We ask them questions like "tell me the First Name of every Employee that's either the Admin of the Sales department or their Manager."

```
FOR     d: Department, e: Employee
WHERE  Name(d)="Sales" 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.

1.4 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, such as 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 direction.

Luckily, the Agda code is pretty easy to understand.

```
record Arena : Set where  -- an arena consists of
  field                     -- two fields
    pos : Set               -- one called pos, a set, and
    dir : pos -> Set        -- one called dir,
                            -- a set for each element of the set pos
```

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 1.19. **Poly** has all products and coproducts and is completely distributive. **Poly** also has exponential objects, making it a biCartesian closed category. It therefore supports the simply typed lambda calculus.

Proof. We will prove that **Poly** has coproducts in Proposition 2.38, that it has products in Proposition 2.65, that it is completely distributive in **??**, and that **Poly** has exponential objects in Theorem 4.28. □

Proposition 1.20. 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.[a] Moreover $\otimes$ is also a closed monoidal structure that distributes over coproducts.

---

[a] We will follow the convention of writing the tensor unit before the tensor product when specifying a monoidal structure.

Proof. We will define $\otimes$ in Definition 2.73 and prove that $(y, \otimes)$ is a monoidal structure on **Poly** in Proposition 2.81, and we will define $\circ$ in **??** and prove that $(y, \circ)$ is a monoidal structure on **Poly** in **??**. Then we will show that $\circ$ is duoidal over $\otimes$ in **??**. In Proposition 2.86, we will show that $\otimes$ distributes over coproducts. Then in Proposition 3.78, we will prove that $\otimes$ is closed. $\square$

Proposition 1.21. **Poly** has an adjoint quadruple with **Set** and an adjoint pair with $\mathbf{Set}^{\mathrm{op}}$:[a]

$$\begin{tikzcd}
\mathbf{Set} \arrow[r, shift left=1, "A" description] \arrow[r, shift right=5, "Ay"'] \arrow[r, phantom, shift left=3, "\Rightarrow"] \arrow[r, phantom, shift right=1, "\Leftarrow"] \arrow[r, phantom, shift right=4, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift right=5, "p(0)"'] \arrow[l, shift left=1, "p(1)" description]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{Set}^{\mathrm{op}} \arrow[r, shift left=2, "y^A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=2, "\Gamma(p)"]
\end{tikzcd} .$$

Each functor is labeled by where it sends $p \in \mathbf{Poly}$ or $A \in \mathbf{Set}$; in particular, $\Gamma(p) := \mathbf{Poly}(p, y)$.

---

[a] We use the notation $C \underset{R}{\overset{L}{\underset{\longleftarrow}{\overset{\longrightarrow}{\Rightarrow}}}} D$ to denote an adjunction $L \dashv R$. The double arrow, always pointing in the same direction as the left adjoint, indicates both the unit $C \Rightarrow R \circ L$ and the counit $L \circ R \Rightarrow D$ of the adjunction.

Proof. We will prove that **Poly** has an adjoint quadruple with **Set** in Theorem 4.4, and that it has an adjoint quadruple with $\mathbf{Set}^{\mathrm{op}}$ in Proposition 4.12. $\square$

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

Proposition 1.22. 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 [Shu08]. But here's where things get really interesting.

Proposition 1.23 (Ahman-Uustalu)**.** Comonoids in $(\mathbf{Poly}, y, \circ)$ are categories (up to isomorphism).

Proposition 1.24. The category $\mathbf{Comon}(\mathbf{Poly})$ has finite coproducts and products, and coproducts in $\mathbf{Comon}(\mathbf{Poly})$ agree with those in **Cat**.

Proposition 1.25. The functor $\mathbf{Comon}(\mathbf{Poly}) \to \mathbf{Poly}$ has a right adjoint

$$\begin{tikzcd}\mathbf{Poly} \arrow[r, shift left, "\mathscr{K}_-"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Comon}(\mathbf{Poly}) \arrow[l, shift left, "u_-"]\end{tikzcd} ,$$

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

Proposition 1.26. 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 1.27. For any polynomial p, the category

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

of dynamical systems on p forms a topos.

The proposition above indicates that there is a full-fledged logic of dynamical systems inhabiting any interface, while the following proposition implies that these logics can be combined and compared.

Proposition 1.28. A morphism $p \to q$ of polynomials induces a pre-geometric morphism between their respective toposes

$$\begin{tikzcd}\mathbf{Sys}(p) \arrow[r, shift left] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Sys}(q) \arrow[l, shift left]\end{tikzcd} .$$

The following propositions suggest that the whole story of dynamics carries a strong connection to database theory.

Proposition 1.29. Suppose that the category $\mathcal{C}$ corresponds under Proposition 1.23 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 1.30 (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 1.31. For any category $\mathcal{C}$ the category of left $\mathscr{C}$-modules is equivalent to the category of functors $\mathcal{C} \to \mathbf{Poly}$,

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

If you skipped over any of that—or all of that—it'll be no problem whatsoever! We will cover each of the above results in detail over the course of this book. There are many avenues for study, but we need to push forward.

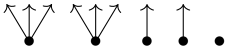
We'll begin in the next chapter.

1.6 Exercise solutions

Solution to Exercise 1.6.

We consider the polynomial $p \coloneqq 2y^3 + 2y + 1$.

1. Here is p drawn as a forest of corollas (note that the order in which the corollas are drawn does not matter):



2. It has five (5) roots.
3. It represents five positions, one per root.
4. The first and second corollas have three leaves, the third and fourth corollas have one leaf, and the fifth corolla has no leaves.
5. The set of directions for each position is the same as the set of leaves for each corolla, so just copy the answer from #4, replacing "corolla" with "position" and "leaf" with "direction." Sheesh! Who wrote this.

Solution to Exercise 1.7.

We refer to the polynomial $q \coloneqq y^8 + 4y$.

1. No, q does not have y^2 as a pure-power summand.
2. Yes, q does have y as a pure-power summand.
3. No, q does not have $4y$ as a pure-power summand, because $4y$ is not a pure-power! But to make amends, we could say that $4y$ is a summand; this means that there is some q' such that $q = q' + 4y$. So $3y$ is also a summand, but $5y$ and y^2 are not.

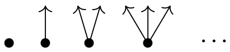
Solution to Exercise 1.8.

Aesthetically speaking, here's the associated polynomial of a beautiful corolla forest:

$$y^0 + y^1 + y^2 + y^3 + \cdots$$

It's reminiscent (and formally related) to the notion of lists: if A is any set, then $A^0 + A^1 + A^2 + \cdots$ is the set of lists (i.e. finite ordered sequences) with entries in A.

Here's a picture of this lovely forest:

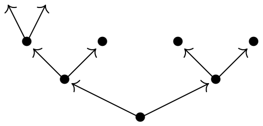


Solution to Exercise 1.11.

When using an application, say a drawing program, there are often buttons I can click at the top of the screen. The information I can send the system changes based on which button I click. If no button is clicked, my options for interacting with the program include clicking any one of the buttons. When I click the "File" button, I can interact with the file system, saving or opening a file, thus interacting with another "part" of the computer. The set of things I can do depends on context, including whatever I did before.

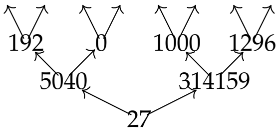
Solution to Exercise 1.15.

1. Here's a level-3 abbreviation of a decision stream of type $y^2 + y^0$.



2. Here's a level-4 abbreviation of a decision stream of type y.

3. Here's a level-3 abbreviation of a decision stream of type $\mathbb{N}y^2$ where we indicate the position of each node by labeling it with a natural number.



Solution to Exercise 1.18.

We refer to (1.17) and (1.16) to characterize the category $\mathcal{C}$ and the functor $I : \mathcal{C} \to \mathbf{Set}$.

1. Here Department is an object of $\mathcal{C}$, so I(Department) is a set. From the Department column in the table on the right of (1.16), we observe that I(Department) = $\{101, 102\}$.
2. Here Admin is an morphism of $\mathcal{C}$ from Department to Employee, so I(Admin) is a function from I(Department) = $\{101, 102\}$ to I(Employee) = $\{1, 2, 3\}$. From the Admin column in the table on the right of (1.16), we observe that I(Admin)(101) = 1 and I(Admin)(102) = 3.
3. By functoriality, I(Admin.FirstName) is the function I(Admin) composed with I(FirstName). We have that I(Admin) sends 101 to 1 and 102 to 3, while the FirstName column tells us that I(FirstName) sends 1 to Alan and 3 to Carla. So I(Admin.FirstName) sends 101 to Alan and 102 to Carla.
4. If every employee works in the same department as their manager, then Employee.WorksIn = Employee.Mngr.WorksIn.
5. The name suggests that I(String) is the set of all possible strings of characters. Perhaps this could be defined as $\bigcup_{n \in \mathbb{N}} A^n$, where A is our alphabet of allowed characters, which may include the English letters, spaces, digits, and whatever other characters we allow. At the very least, since FirstName and Name are both morphisms to String, every entry in the FirstName and Name columns must be in I(String). So all we know for sure is that {Alan, Ruth, Carla, Sales, IT} $\subseteq$ I(String).

Chapter 2

Polynomial functors and natural transformations

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.

2.1 Representable functors and the Yoneda lemma

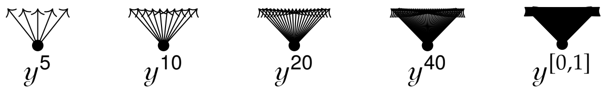
Definition 2.1. Given any set S, we denote by $y^S \colon \mathbf{Set} \to \mathbf{Set}$ the functor that sends any set X to the set $X^S = \mathbf{Set}(S, X)$, and sends any function $h \colon X \to Y$ to the function $h^S \colon X^S \to Y^S$, the one that sends $g \colon S \to X$ to $g \mathbin{⨾} h \colon S \to Y$.[a]

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

---

[a] Throughout this text, in any category, given objects A, B, C and morphisms $f \colon A \to B$ and $g \colon B \to C$, we denote their composite morphism $A \to C$ as $f \mathbin{⨾} g$ (instead of the usual $g \circ f$).

The symbol y stands for Yoneda, for reasons we will get to in Lemma 2.10. For now, here are some pictures for your eyes to gaze at; they are the polys corresponding to various representables, namely the pure powers:



(2.2)

Example 2.3. The functor that sends every set X to $X \times X$, and sends $h \colon X \to Y$ to $(h \times h) \colon (X \times X) \to (Y \times Y)$, is representable. After all, $X \times X \cong X^2$, so this functor is just a pure power we are already familiar with: y^2.

Exercise 2.4 (Solution here). For each of the following functors **Set** → **Set**, say if it is representable or not; if it is, say what the representing set is.

1. The identity functor $X \mapsto X$, which sends each function to itself.
2. The constant functor $X \mapsto \mathsf{2}$, which sends every function to the identity on $\mathsf{2}$.
3. The constant functor $X \mapsto \mathsf{1}$, which sends every function to the identity on $\mathsf{1}$.
4. The constant functor $X \mapsto \mathsf{0}$, which sends every function to the identity on $\mathsf{0}$.
5. A functor $X \mapsto X^{\mathbb{N}}$. If it were representable, where would it send each function?
6. A functor $X \mapsto 2^X$. If it were representable, where would it send each function?

◊

Proposition 2.5. For any function $f\colon R \to S$, there is an induced natural transformation $y^f\colon y^S \to y^R$; on any set X the X-component $X^f\colon X^S \to X^R$ is given by sending $g\colon S \to X$ to $f \mathbin{⨟} g\colon R \to X$.

Exercise 2.6 (Solution here). Prove that for any function $f\colon R \to S$, what we said was a natural transformation in Proposition 2.5 really is natural. That is, for any function $h\colon X \to Y$, show that the following diagram commutes:

$$\begin{tikzcd}
X^S \arrow[r, "h^S"] \arrow[d, "X^f"'] \arrow[dr, phantom, "?"] & Y^S \arrow[d, "Y^f"] \\
X^R \arrow[r, "h^R"'] & Y^R
\end{tikzcd}$$

◊

Exercise 2.7 (Solution here). Let X be an arbitrary set. For each of the following sets R, S and functions $f\colon R \to S$, describe the X-component of, i.e. the function $X^S \to X^R$ coming from, the natural transformation $y^f\colon y^S \to y^R$.

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

◊

Exercise 2.8 (Solution here). Show that the construction in Proposition 2.5 is functorial

$$y^{-}\colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}^{\mathbf{Set}}, \tag{2.9}$$

as follows.

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

Lemma 2.10 (Yoneda lemma)**.** Given a functor $F\colon \mathbf{Set} \to \mathbf{Set}$ and a set S, there is an isomorphism

$$F(S) \cong \mathsf{Nat}(y^S, F) \tag{2.11}$$

where Nat denotes the set of natural transformations. Moreover, (2.11) is natural in both S and F.

Sketch of proof. For any natural transformation $m\colon y^S \to F$, consider the component $m_S\colon S^S \to F(S)$. 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 $m^a\colon y^S \to F$ whose X-component is the function $X^S \to F(X)$ given by sending $g\colon S \to X$ to $F(g)(a)$. In Exercise 2.12 we ask you to show that this is natural in X and that these two constructions are mutually inverse. □

Exercise 2.12 (Solution here). 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 the proof sketch of Lemma 2.10 are natural in X.
2. Show that the two mappings from the proof sketch of Lemma 2.10 are mutually inverse.
3. Show that (2.11) is natural in F.
4. Show that (2.11) is natural in S.
5. As a corollary of Lemma 2.10, show that $y^-\colon \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}^{\mathbf{Set}}$ is fully faithful, in particular that there is an isomorphism $\mathsf{Nat}(y^S, y^T) \cong S^T$. ◊

2.2 Polynomials: sums of representables

We've seen that for any set A, the symbol y^A represents a functor $\mathbf{Set} \to \mathbf{Set}$. 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 the term "polynomial" in this way.

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.[1] For example, in the

---

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

following bizarre polynomial

$$p \coloneqq \mathbb{R}y^{\mathbb{Z}} + \mathsf{3}y^3 + Ay + \sum_{i \in I} Q_i y^{R_i + Q_i^2}, \tag{2.13}$$

$\mathbb{R}$ denotes the set of real numbers, $\mathbb{Z}$ denotes the set of integers, $\mathsf{3}$ 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 (2.13) 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 $y^A \colon \mathbf{Set} \to \mathbf{Set}$. So to understand general polynomials like p, we just need to understand what the sums of functors are. It will be useful to discuss products of functors at the same time.

To undertand the sums and products of set-valued functors, we will first need to understand the sums and products of sets.

2.2.1 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* — in the form of a functor as $X \colon I \to \mathbf{Set}$ (where we view the set I as a discrete category) or more classically as $(X_i)_{i \in I}$. Indeed, when a functor $X \colon I \to \mathbf{Set}$ is understood to be a dependent set, we will denote $X(i)$ as X_i for $i \in I$. 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

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

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

Definition 2.14 (Dependent sums and products of sets)**.** Let I be a set and $X \colon I \to \mathbf{Set}$ be an I-indexed collection of sets. The *sum* $\sum_{i \in I} X_i$ and *product* $\prod_{i \in I} X_i$ of this collection are the sets

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

Example 2.15. If $I = \mathsf{2} = \{1, 2\}$ then a collection $X \colon I \to \mathbf{Set}$ is just two sets, say $X_1 = \{a, b, c\}$ and $X_2 = \{c, d\}$. Their sum is the disjoint union

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

Its cardinality (i.e. the number of elements it contains) will always be the sum of the cardinalities of X_1 and X_2.

Meanwhile, their product is the usual Cartesian product

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

Its cardinality will always be the product of the cardinalities of X_1 and X_2.

Exercise 2.16 (Solution here). Let I be a set and let $X_i := \mathsf{1}$ be a one-element set for each $i \in I$.

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

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

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

Exercise 2.17 (Solution here). Let $X \colon I \to \mathbf{Set}$ be a set depending on an $i \in I$. There is a projection function $\pi_1 \colon \sum_{i \in I} X_i \to I$ defined by $\pi_1(i, x) = i$.

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

$$\prod_{i \in I} X_i \cong \left\{ s \colon I \to \sum_{i \in I} X_i \;\middle|\; s \mathbin{⨾} \pi_1 = \mathrm{id}_I \right\}.$$

◊

A helpful way to think about sum or product sets is by considering what choices must be made to specify an element of such a set. In the following examples, say that we have a dependent set $X \colon I \to \mathbf{Set}$.

Here we give the instructions for choosing an element of $\sum_{i \in I} X_i$.

> To choose an element of $\sum_{i \in I} X_i$:
>
> 1. choose an element $i \in I$;
> 2. choose an element of X_i.

Then the projection π_1 from Exercise 2.17 sends each element of $\sum_{i \in I} X_i$ to the element of $i \in I$ chosen in step 1, while the projection π_2 sends each element of $\sum_{i \in I} X_i$ to the element of X_i chosen in step 2.

Now we give the instructions for choosing an element of $\prod_{i \in I} X_i$.

> To choose an element of $\prod_{i \in I} X_i$:
>
> 1. for each element $i \in I$:
>   1. choose an element of X_i.

Armed with these interpretations, we can tackle more complicated expressions, including those with nested $\sum$'s and $\prod$'s like

$$A \coloneqq \sum_{i \in I} \prod_{j \in J(i)} \sum_{k \in K(i,j)} X(i,j,k). \tag{2.18}$$

We can give the instructions for choosing an element of A as a nested list, as follows.

> To choose an element of A:
>
> 1. choose an element $i \in I$;
> 2. for each element $j \in J(i)$:
>   1. choose an element $k \in K(i,j)$;
>   2. choose 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 different values of i and j may lead to different sets $K(i,j)$.

By describing A like this, it is clear that each $a \in A$ can be projected to an element $\pi_1(a) \in I$ from step 1 and a dependent function $\pi_2(a)$ from step 2. This dependent function in turn sends each $j \in J(i)$ to a pair that can be projected to an element $\pi_1(\pi_2(a)(j)) \in K(i,j)$ from step 2.1 and an element $\pi_2(\pi_2(a)(j)) \in X(i,j,k)$ from step 2.2.

Example 2.19. Let $I = \{1, 2\}$, let $J(1) = \{j\}$ and $J(2) \coloneqq \{j, j'\}$, let $K(1,j) \coloneqq \{k_1, k_2\}$, $K(2,j) \coloneqq \{k_1\}$, and $K(2,j') \coloneqq \{k'\}$, and let $X(i,j,k) = \{x, y\}$ for all i, j, k. Now the formula

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

from (2.18) has been given meaning as an actual set. Here is a list of all eight of its elements:

$$\left\{\begin{array}{cccc} \big(1, j \mapsto (k_1, x)\big), & \big(1, j \mapsto (k_1, y)\big), & \big(1, j \mapsto (k_2, x)\big), & \big(1, j \mapsto (k_2, y)\big), \\ & \big(2, j \mapsto (k_1, x), j' \mapsto (k', x)\big), & \big(2, j \mapsto (k_1, x), j' \mapsto (k', y)\big), & \\ & \big(2, j \mapsto (k_1, y), j' \mapsto (k', x)\big), & \big(2, j \mapsto (k_1, y), j' \mapsto (k', y)\big) & \end{array}\right\}$$

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 2.20 (Solution here). Consider the set

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

1. Give the instructions for choosing an element of B as a nested list, like we did for A just below (2.18).
2. With I, J, K, and X as in Example 2.19, how many elements are in B?
3. Write out three of these elements in the style of Example 2.19. ◊

2.2.2 Expanding products of sums

We will often encounter sums of dependent sets nested within products. The following proposition helps us work with these; it 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; in particular, once the statement is written in Agda, its proof is one short line of Agda code.

Proposition 2.21 (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

$$\prod_{i\in I}\sum_{j\in J(i)} X(i,j) \cong \sum_{\bar{j}\in\prod_{i\in I}J(i)}\prod_{i\in I} X(i,\bar{j}(i)). \tag{2.22}$$

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

First, let's go from left to right. An element of the set on the left is a dependent function $f\colon (i\in I)\to\sum_{j\in J(i)} X(i,j)$, which we can compose with projections from its codomain to yield $\pi_1(f(i))\in J(i)$ and $\pi_2(f(i))\in X(i,\pi_1(f(i)))$ for every $i\in I$. We can then form the following pair in the right hand set:

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

Now let's go from right to left. An element of the set on the right is a pair of dependent functions, $\bar{j}\colon (i\in I)\to J(i)$ and $g\colon (i\in I)\to X(i,\bar{j}(i))$. We then get an element of the set on the left as follows:

$$i\mapsto(\bar{j}(i), g(i)).$$

Now, we just need to check that a round trip takes us back where we were. If we start on the right from $(\bar{j}, g)$, our round trip gives us the pair

$$(i\mapsto\pi_1(\bar{j}(i), g(i)), i\mapsto\pi_2(\bar{j}(i), g(i))).$$

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

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

But again, since $f(i)$ is a pair whose components are $\pi_1(f(i))$ and $\pi_2(f(i))$, we're back where we started.

□

When $J(i) = J$ does not depend on $i \in I$, the formula in (2.22) becomes much easier.

Corollary 2.23. For any set I, set J, and collection of sets $\{X(i,j)\}_{i\in I, j\in J}$, we have a bijection

$$\prod_{i\in I}\sum_{j\in J} X(i,j) \cong \sum_{\bar{j}\colon I\to J}\prod_{i\in I} X(i,\bar{j}(i)). \tag{2.24}$$

Proof. Just take $J(i) = J$ for all $i \in I$ in (2.22). Note that dependent functions $\bar{j}$ in $\prod_{i\in I} J(i)$ then become standard functions $\bar{j}\colon I \to J$. □

Below, e.g. in Exercise 2.48, you'll often see alternating products and sums; using (2.22), you can always write it as a sum of products, in which every $\sum$ appears before every $\prod$ (i.e. in "disjunctive normal form"). This is analogous to how products of sums in high school algebra can always be expanded into sums of products via the distributive property.

2.2.3 Dependent sums and products of functors Set → Set

Where are we, and where are we going? We've defined 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 **Set** → **Set**. But we might as well define products of functors at the same time, because they'll very much come in handy.

Definition 2.25 (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

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

We may also denote the product functor $F \times G$ by FG.

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

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

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

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

Given a set $I \in \mathbf{Set}$, we will also use I to denote the constant functor that assigns I to each $X \in \mathbf{Set}$. In particular, we denote by $\mathsf{0}, \mathsf{1} \colon \mathbf{Set} \to \mathbf{Set}$ the constant functors that respectively assign $\mathsf{0}$ and $\mathsf{1}$ to each $X \in \mathbf{Set}$.

Exercise 2.26 (Solution here).

1. Show that for a set $I \in \mathbf{Set}$ and a functor $F \colon \mathbf{Set} \to \mathbf{Set}$, the sum of I copies of F is isomorphic to the product of the constant functor I and F:

   $$\sum_{i\in I} F \cong IF.$$

   (Note that this is analogous to the fact from basic arithmetic that adding up $n \in \mathbb{N}$ copies of number is equal to multiplying that same number by n.)
2. So does $\mathsf{2}y$ denote $\mathsf{2} \times y$ or $y + y$, or does it not matter for some reason? ◊

Proposition 2.27. Referring to the notation in Definition 2.25, the functors $\mathsf{0}$ and $\mathsf{1}$ 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 2.15 and Exercise 2.16, 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 $\mathbf{Set}$) has limits and colimits; see [MM92, page 22 – 23, displays (24) and (25)]. □

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

2.2.4 What is a polynomial functor?

Definition 2.28 (Polynomial functors)**.** A *polynomial functor* (or simply a *polynomial*) 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.

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

Remark 2.29. Given sets $I, A \in \mathbf{Set}$, it follows from Exercise 2.26 that we have an isomorphism of polynomials

$$\sum_{i\in I} y^A \cong Iy^A.$$

So when we write down a polynomial, we will often combine identical representable summands y^A by writing them in the form Iy^A. In particular, the constant functor $\mathsf{1}$ is a representable functor ($\mathsf{1} \cong y^{\mathsf{0}}$), so every constant functor I is a polynomial functor: $I \cong \sum_{i\in I} \mathsf{1}$.

Example 2.30. Consider the polynomial $p := y^2 + 2y + 1$. It denotes a functor $\mathbf{Set} \to \mathbf{Set}$; what does this functor do to the set $X := \{a, b\}$? To be very precise and pedantic, let's say

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

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

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

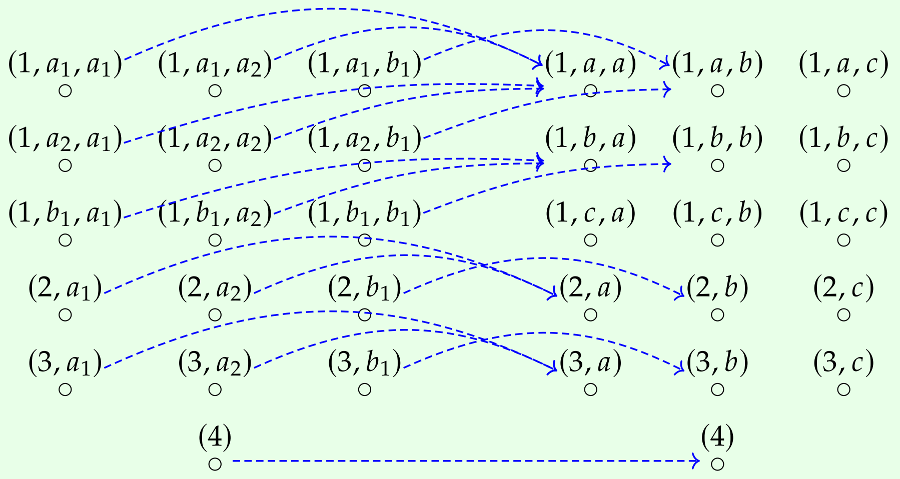
It has $(2^2 + 2 + 2 + 1)$-many, i.e. 9, elements. The representable summand y^A throws in all A-tuples from X, but it's indexed by the name of the summand. In particular if $A = \mathsf{0}$ then it just records the empty tuple at that summand.

Exercise 2.31 (Solution here). In the pedantic style of Example 2.30, write out all the elements of $p(X)$ for p and X as follows:

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

Example 2.32. Suppose $p := y^2 + 2y + 1$. As a functor $\mathbf{Set} \to \mathbf{Set}$ 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 $f\colon X \to Y$ be the function sending $a_1, a_2 \mapsto a$ and $b_1 \mapsto b$. The induced function $p(f)\colon p(X) \to p(Y)$

is shown below



Exercise 2.33 (Solution here). Let $p \coloneqq y^2 + y$. Choose a function $f\colon \mathsf{1} \to \mathsf{2}$ and write out the induced function $p(f)\colon p(\mathsf{1}) \to p(\mathsf{2})$. ◊

Proposition 2.34. Let $p \coloneqq \sum_{i \in I} y^{p[i]}$ be an arbitrary polynomial functor. Then $I \cong p(\mathsf{1})$, so there is an isomorphism of functors

$$p \cong \sum_{i \in p(\mathsf{1})} y^{p[i]}. \tag{2.35}$$

Proof. We need to show that $I \cong p(\mathsf{1})$; the latter claim follows directly. In Exercise 2.16 it was shown that $I \cong \sum_{i \in I} \mathsf{1}$, so we just need to show that $(y^{p[i]})(\mathsf{1}) \cong \mathsf{1}$ for every $i \in I$. But $\mathsf{1}^{p[i]} \cong \mathsf{1}$ because there is a unique function $p[i] \to \mathsf{1}$ for any $p[i]$. □

We can draw an analogy between Proposition 2.34 and evaluating $p(1)$ for a polynomial p from high school algebra, which yields the sum of the coefficients of p. The notation in (2.35) will be how we denote arbitrary polynomials from now on.

Exercise 2.36 (Solution here). We saw in Proposition 2.34 that for any polynomial p, e.g. $p \coloneqq y^3 + 3y^2 + 4$, the set $p(\mathsf{1})$ gives back the set of summands, in this case $\mathsf{8}$.

What does $p(\mathsf{0})$ give you? ◊

2.3 Morphisms between polynomial functors

Before we define the category **Poly** of polynomial functors, we notice that polynomial functors already live inside a category, namely the category $\mathbf{Set}^{\mathbf{Set}}$ of functors $\mathbf{Set} \to \mathbf{Set}$, whose morphisms are natural transformations. This leads to a very natural (if you will) definition of morphisms between polynomial functors, from which we can derive a category of polynomial functors for free.

Definition 2.37 (Polynomial morphisms, **Poly**)**.** Given polynomial functors p and q, a *morphism of polynomial functors* (or a *polynomial morphism*) is a natural transformation $p \to q$. Then **Poly** is the category whose objects are polynomial functors and whose morphisms are polynomial morphisms.

In other words, **Poly** is the full subcategory of $\mathbf{Set}^{\mathbf{Set}}$ spanned by the polynomial functors: we take the category $\mathbf{Set}^{\mathbf{Set}}$, throw out all the objects that are not polynomials, but keep all the same morphisms between any two polynomial functors.

2.3.1 Coproducts of polynomials

Since polynomial functors are defined as arbitrary sums of representables, coproducts in **Poly** are quite easy to understand.

Proposition 2.38. The category **Poly** has arbitrary coproducts, given by the operation $\sum_{i \in I}$.

Proof. By Proposition 2.27, the category $\mathbf{Set}^{\mathbf{Set}}$ has arbitrary coproducts given by $\sum_{i \in I}$. The full subcategory inclusion $\mathbf{Poly} \to \mathbf{Set}^{\mathbf{Set}}$ reflects these coproducts, and by definition **Poly** is closed under the operation $\sum_{i \in I}$. □

Explicitly, given polynomials $\{p_i\}_{i \in I}$, their coproduct is

$$\sum_{i \in I} p_i \cong \sum_{i \in I} \sum_{j \in p_i(1)} y^{p_i[j]} \cong \sum_{(i,j) \in \sum_{i \in I} p_i(1)} y^{p_i[j]}, \tag{2.39}$$

which coincides with our notion of sums of functors $\mathbf{Set} \to \mathbf{Set}$ from Definition 2.25. Binary coproducts are thus given by binary sums of functors, denoted by $+$. In particular, (2.39) implies that for any polynomials p and q, their coproduct $p + q$ is given as follows. The set of positions of $p + q$ is the coproduct of sets $p(1) + q(1)$. At position $(1, i) \in p(1) + q(1)$ with $i \in p(1)$, the directions of $p + q$ are just the directions of p at i; at position $(2, j) \in p(1) + q(1)$ with $j \in q(1)$, the directions of $p + q$ are just the directions of q at j.

2.3.2 Polynomial morphisms, concretely

A natural transformation between polynomials $p \to q$ consists of a function $p(X) \to q(X)$ for every set $X \in \mathbf{Set}$ such that the naturality squares commute. That's a lot of data to keep track of! Fortunately, there is a much simpler way to think about these polynomial morphisms, which we will discover with some help from our old friend, the Yoneda lemma.

Exercise 2.40 (Solution here). Given a set S and a polynomial q, show that a polynomial morphism $y^S \to q$ can be identified with an element of the set $q(S)$. That is, there is an isomorphism

$$\mathbf{Poly}(y^S, q) \cong q(S).$$

Moreover, show that this isomorphism is natural in both S and q. Hint: Use the Yoneda lemma (Lemma 2.10). ◊

Before we present our alternative characterization of polynomial morphisms, recall that every polynomial $p \coloneqq \sum_{i \in p(1)} y^{p[i]}$ is uniquely associated to a dependent set, $(p[i])_{i \in p(1)}$, which we call its *arena*, as in (1.3). Alternatively, we could write such a dependent set as a functor $p[-]\colon p(1) \to \mathbf{Set}$, where we view the set $p(1)$ as a discrete category. Below, we make use of this functor notation to express the arenas of polynomials.

Proposition 2.41. Let $p \coloneqq \sum_{i \in p(1)} y^{p[i]}$ and $q \coloneqq \sum_{j \in q(1)} y^{q[j]}$ be polynomials. Then we have an isomorphism

$$\mathbf{Poly}(p, q) \cong \prod_{i \in p(1)} \sum_{j \in q(1)} p[i]^{q[j]}. \tag{2.42}$$

In other words, a morphism $p \to q$ can be identified with a pair $(f_1, f^\sharp)$

$$\begin{tikzcd} p(1) \arrow[rr, "f_1"] \arrow[dr, "{p[-]}"'] & \arrow[d, phantom, "\overset{f^\sharp}{\Leftarrow}"] & q(1) \arrow[dl, "{q[-]}"] \\ & \mathbf{Set} & \end{tikzcd} \tag{2.43}$$

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

Proof. By the universal property of the coproduct, we have an isomorphism

$$\mathbf{Poly}\left(\sum_{i \in p(1)} y^{p[i]}, q\right) \cong \prod_{i \in p(1)} \mathbf{Poly}(y^{p[i]}, q),$$

so applying Exercise 2.40 (i.e. the Yoneda lemma) and unraveling the definitions of p and q yields (2.42).

The expression on the right hand side of (2.42) is the set of dependent functions $f\colon (i \in p(1)) \to \sum_{j \in q(1)} p[i]^{q[j]}$, each of which is uniquely determined by its components: $\pi_1 \circ f$, which sends $i \in p(1)$ to $\pi_1(f(i)) \in q(1)$, and $\pi_2 \circ f$, which sends $i \in p(1)$ with $j \coloneqq \pi_1(f(i))$ to an element of $p[i]^{q[j]}$, i.e. a function $q[j] \to p[i]$. These can be identified respectively with a (non-dependent) function $f_1 \coloneqq \pi_1 \circ f$ from $p(1) \to q(1)$ and a natural transformation $f^\sharp\colon q[f_1(-)] \to p[-]$. □

We have now greatly simplified our characterization of polynomial morphisms $f\colon p \to q$: rather than infinitely many functions satisfying infinitely many naturality conditions, they can be specified simply as a function $f_1\colon p(1) \to q(1)$ and, for each $i \in p(1)$, a function $f_i^\sharp\colon q[f_1(i)] \to p[i]$, without any additional restrictions.

Here is where we begin to see the advantages of viewing polynomials as arenas. As a reminder, from the arena perspective, we call the elements of $p(1)$ the *positions* of p, and for each position $i \in p(1)$, we call the elements of $p[i]$ the *directions* of p at i. We can see that our characterization of a polynomial morphism can be written entirely in the language of positions and directions: when polynomials p and q are viewed as arenas, a morphism $f\colon p \to q$ consists of a "forwards" *on-positions* function f_1 from the positions of p to the positions of q, along with, for every position i of p, a "backwards" *on-directions* function $f_i^\sharp$ from the directions of q at $f_1(i)$ to the directions of p at i.

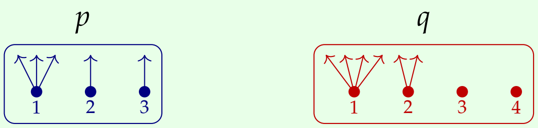
When we wish to emphasize the arena perspective, we will call the data of $(f_1, f^\sharp)$ a *morphism of arenas* or *arena morphism* between p and q; but since Proposition 2.41 tells us that polynomial morphisms and arena morphisms carry the same data, we may use these terms interchangeably.

Here is how you would implement such a morphism in Agda:

```
record ArenaMorphism (p : Arena) (q : Arena) : Set where
  field
    onPos : (pos p) -> (pos q)
    onDir : (i : pos p) -> dir q (onPos i) -> dir p i
```

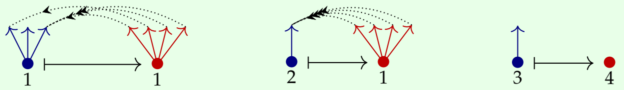
This forwards on-positions/backwards on-directions formulation may still seem a little complicated, so here is some decision-making intuition. Recall from Section 1.2 that we may view each position of an arena as a decision and the directions at that position as the options available for that decision. The morphisms $f\colon p \to q$ 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.

Example 2.44. Let $p \coloneqq y^3 + 2y$ and $q \coloneqq y^4 + y^2 + 2$. Here they are, depicted as corolla forests:



To give a map of polynomials $p \to q$, one sends each position $i \in p(1)$ of p to a position $j \in q(1)$ of q, then sends each direction in $q[j]$ 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? The first position of p can be sent to any position of q: 1, 2, 3, or 4. Sending it to 1 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 handle the first position of p: there are $3^4 + 3^2 + 3^0 + 3^0 = 92$.

The second position of p can also be sent to 1, 2, 3, or 4, before sending back directions; there are $1^4 + 1^2 + 1^0 + 1^0 = 4$ ways to do this. Similarly there are four ways to send the third position of p to a position of q and send back directions.

In total, there are $92 \cdot 4 \cdot 4 = 1472$ morphisms $p \to q$.

Unsurprisingly, this is exactly what is given by (2.42):

$$\begin{aligned}
|\mathbf{Poly}(p, q)| &= \prod_{i \in p(1)} |q(p[i])| \\
&= \prod_{i \in p(1)} |p[i]|^4 + |p[i]|^2 + 2 \\
&= (3^4 + 3^2 + 2)(1^4 + 1^2 + 2)^2 \\
&= 92 \cdot 4^2 = 1472.
\end{aligned}$$

Exercise 2.45 (Solution here).

1. Draw the corolla forests associated to $p := y^3 + y + 1$, $q := y^2 + y^2 + 2$, and $r := y^3$.
2. Give an example of a morphism $p \to q$ and draw it as we did in Example 2.44.
3. Explain your morphism intuitively as a delegation of decisions.
4. Explain in those terms why there can't be any morphisms $p \to r$. ◊

Exercise 2.46 (Solution here). 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 both y^2 and p as corolla forests.
2. Count all the maps $y^2 \to p$. How many are there?
3. Is the previous answer equal to $p(2)$? ◊

Exercise 2.47 (Solution here). For each of the following polynomials p, q, compute the number of morphisms $p \to q$.

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 2.48 (Solution here).

1. Is it true that the following are isomorphic?

$$\mathbf{Poly}(p, q) \cong^? \prod_{i \in p(1)} \sum_{j \in q(1)} \prod_{d \in q[j]} \sum_{c \in p[i]} 1 \tag{2.49}$$

2. Is it true that the following are isomorphic?

$$\mathbf{Poly}(p, q) \cong^? \sum_{f_1 \colon p(1) \to q(1)} \prod_{j \in q(1)} \mathbf{Set}\left(q[j], \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]\right) \tag{2.50}$$

3. If the answer to #2 is "yes," then describe in the language of decision-making 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. ◊

Exercise 2.51 (Solution here). Use (2.39) and (2.42) to verify that

$$\mathbf{Poly}\left(\sum_{i \in I} p_i, q\right) \cong \prod_{i \in I} \mathbf{Poly}(p_i, q)$$

for all polynomials $\{p_i\}_{i \in I}$ and q, as expected from the universal property of coproducts. ◊

Example 2.52 (Derivatives). The *derivative* of a polynomial p, denoted $\dot{p}$, is defined as follows:

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

For example, if $p := y^{\{U,V,W\}} + \{A, B\}y^{\{X\}}$ 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 $\dot{p} \cong 3y^2 + 2$. Unsurprisingly, this coincides with the familiar notion of derivatives of polynomials from calculus.

Thus we get a canonical map $\dot{p}y \to p$, because we have an isomorphism

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

This natural transformation comes up in computer science in the context of "plugging in to one-hole contexts"; we will not explore that here, but see [**mcbride**] and [**abbot2003derivatives**] for more info.

A morphism $f\colon p \to \dot{q}$ can be interpreted as something like an arena morphism from p to q, except that each position of p explicitly selects a direction of q to remain unassigned. 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 position j of q, as usual, together with a chosen direction $d \in q[j]$. Then every direction of q at j *other than* d is sent back to an direction of p at i.

Exercise 2.53 (Solution here). Show that $\dot{p}(1)$ is isomorphic to the set of all directions of p (i.e. the union of all direction-sets of p), so there is a canonical function $\pi_p\colon \dot{p}(1) \to p(1)$ that sends each direction d of p to the position i of p for which $d \in p[i]$. ◊

Exercise 2.54 (Solution here). The derivative is not very well-behaved category-theoretically. However, it is intriguing. Below $p, q \in$ **Poly**.

1. Explain the canonical map $\dot{p}y \to p$ from Example 2.52 in more detail.
2. Is there always a canonical map $p \to \dot{p}$?
3. Is there always a canonical map $\dot{p} \to p$?
4. If given a map $p \to q$, does one get a map $\dot{p} \to \dot{q}$?
5. We will define the binary operations $\otimes$ and $[-,-]$ on **Poly** later on in (2.74) and (3.68), and in Exercise 3.73, you will be able to use Exercise 3.71 to deduce that

$$[p, y] \otimes p \cong \sum_{f \in \prod_{i \in p(1)} p[i]} \sum_{i \in p(1)} y^{p(1) \times p[i]}, \tag{2.55}$$

   Is there always a canonical map $[p, y] \otimes p \to \dot{p}$?
6. When talking to someone who explains maps $p \to \dot{q}$ in terms of "unassigned directions," how might you describe what is modeled by a map $py \to q$? ◊

2.3.3 Translating between natural transformations and arena morphisms

We now know that we can specify a morphism of polynomials $p \to q$ in two ways: in the language of functors, by specifying a natural transformation from p to q; or in the language of arenas, by specifying a function $f_1\colon p(1) \to q(1)$ and, for each $i \in p(1)$, a function $f_i^\sharp\colon q[f_1(i)] \to p[i]$. But what is the relationship between these two formulations? If you told me an arena morphism, and I told you a natural transformation between polynomial functors, how could we tell if we were talking about the same morphism or not? We want to be able to translate between these two languages.

Our Rosetta Stone turns out to be the proof of the Yoneda lemma. The lemma itself forms the crux of the proof of Proposition 2.41, that these two formulations of polynomial morphisms are equivalent; so unraveling this proof reveals the translation we seek.

Proposition 2.56. Let p and q be polynomial functors, and let $(f_1, f^\sharp)$ be a morphism between their associated arenas. Then the isomorphism in (2.42) sends $(f_1, f^\sharp)$ to the natural transformation $f\colon p \to q$ whose X-component $f_X\colon p(X) \to q(X)$ for each $X \in \mathbf{Set}$ sends every

$$(i, g) \in \sum_{i \in p(1)} X^{p[i]} \cong p(X),$$

with $i \in p(1)$ and $g\colon p[i] \to X$, to

$$(f_1(i), f_i^\sharp \mathbin{⨟} g) \in \sum_{j \in q(1)} X^{q[j]} \cong q(X).$$

Proof. As an element of the product over I on the right hand side of (2.42), the pair $(f_1, f^\sharp)$ can equivalently be thought of as a set of pairs $\{(f_1(i), f_i^\sharp)\}_{i \in I}$. Fixing $i \in I$, the pair $(f_1(i), f_i^\sharp)$ is an element of

$$\sum_{j \in q(1)} p[i]^{q[j]} = q(p[i])$$

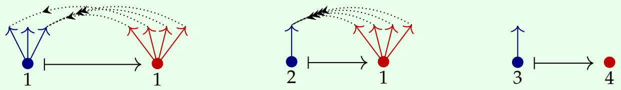
(so $f_1(i) \in q(1)$ and $f_i^\sharp\colon q[f_1(i)] \to p[i]$). By the Yoneda lemma (Lemma 2.10), we have an isomorphism $q(p[i]) \cong \mathbf{Poly}(y^{p[i]}, q)$, and by the proof of the Yoneda lemma, this isomorphism sends $(f_1(i), f_i^\sharp)$ to the natural transformation $f^i\colon y^{p[i]} \to q$ whose X-component is the function $f_X^i\colon X^{p[i]} \to q(X)$ given by sending $g\colon p[i] \to X$ to

$$q(g)(f_1(i), f_i^\sharp) = \left(\sum_{j \in q(1)} g^{q[j]}\right)(f_1(i), f_i^\sharp) = \left(f_1(i), g^{q[f_1(i)]}(f_i^\sharp)\right) = (f_1(i), f_i^\sharp \mathbin{⨟} g).$$

Taken together, the collection of natural transformations $\{f^i\}_{i \in I}$ is an element of $\prod_{i \in I} \mathbf{Poly}(y^{p[i]}, q)$. Applying the universal property of coproducts, as in the proof of Proposition 2.41, we find that $\{f^i\}_{i \in I}$ corresponds to the natural transformation $f\colon p \to q$ we desire. □

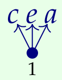
Example 2.57. Let us return to the polynomials $p := y^3 + 2y$ and $q := y^4 + y^2 + 2$ from

Example 2.44 and the morphism $f\colon p \to q$ depicted below:



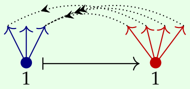
Fix a set $X \coloneqq \{a, b, c, d, e\}$. When viewed as a natural transformation, the morphism f has as its X-component a function $f_X\colon p(X) \to q(X)$. In other words, for any element of $p(X)$, the morphism f should be able to give us an element of $q(X)$.

What does an element of $p(X)$ look like? Well, to specify such an element, we would need to choose a position i of p and a function $p[i] \to X$. We can depict this by selecting one of the corollas in the forest of p and labeling each leaf of that corolla with an element of X. For example, here we depict an element $(1, g)$ of $p(X)$, where $g\colon p[1] \to X$ is given by $1 \mapsto c, 2 \mapsto e$, and $3 \mapsto a$:



Similarly, an element of $q(X)$ can be drawn as a corolla in the forest of q, with each leaf labeled by an element of X. So what element of $q(X)$ is $f_X(1, g)$?

Proposition 2.56 tells us that $f_X(1, g)$ can be read off of the forest depiction of f at position 1:



To draw $f_X(1, g)$, we first draw the corolla in the forest of q corresponding to $f_1(1)$: the corolla on the right hand side above. Then we label each leaf of that corolla by following the arrow from that leaf (as given by $f_i^\sharp$) to a leaf of p at 1, and use the label there that is given by $(1, g)$. So $f_X(1, g)$ looks like

Proposition 2.56 lets us translate from arena morphisms to natural transformations. The following corollary tells us how to go in the other direction. In particular, it justifies the notation f_1 for the on-positions function of f.

Corollary 2.58. Let p and q be polynomial functors, and let $f\colon p \to q$ be a natural transformation between them. Then the isomorphism in (2.42) sends f to the arena morphism $(f_1, f^\sharp)$ for which $f_1\colon p(1) \to q(1)$ is the 1-component of f and, for each $i \in p(1)$, we have

$$(f_1(i), f_i^\sharp) = f_{p[i]}(i, \mathrm{id}_{p[i]}).$$

Proof. By Proposition 2.56, the 1-component $f_1\colon p(1) \to q(1)$ sends every $i \in p(1)$ to $f_1(i) \in q(1)$, so the on-positions function f_1 is equal to the 1-component f_1. Also, the

$p[i]$-component $f_{p[i]}\colon p(p[i]) \to q(p[i])$ sends every $(i, \mathrm{id}_{p[i]}) \in p(p[i])$, with $i \in p(1)$, to $(f_1(i), f_i^\sharp \fatsemi \mathrm{id}_{p[i]}) = (f_1(i), f_i^\sharp)$. □

2.3.4 Identity and composition of arena morphisms

Thus far, we have seen how the category **Poly** of polynomial functors and natural transformations can just as easily be thought of as the category of arenas and arena morphisms. But in order to actually discuss the latter category, we need to be able to give identity arena morphisms and describe how these arena morphisms compose. To do so, we can leverage our ability to translate back and forth between arena morphisms and natural transformations.

For instance, given a polynomial p, the identity morphism of its associated arena should correspond to the identity natural transformation of p as a functor.

Exercise 2.59 (Identity arena morphisms; solution here). Let p be a polynomial and let $\mathrm{id}_p\colon p \to p$ be its identity natural transformation, whose X-component $(\mathrm{id}_p)_X\colon p(X) \to p(X)$ for each $X \in \mathbf{Set}$ is the identity function on $p(X)$; that is, $(\mathrm{id}_p)_X = \mathrm{id}_{p(X)}$.

Use Corollary 2.58 to show that the arena morphism $((\mathrm{id}_p)_1, (\mathrm{id}_p)^\sharp)$ associated to id_p is such that $(\mathrm{id}_p)_1\colon p(1) \to p(1)$ and $(\mathrm{id}_p)_i^\sharp\colon p[(\mathrm{id}_p)_1(i)] \to p[i]$ for $i \in p(1)$ are all identity functions. ◊

Similarly, we should be able to deduce how two arena morphisms compose by translating them to natural transformations, composing those, then translating back to arena morphisms.

Exercise 2.60 (Composing arena morphisms; solution here). Let p, q, and r be polynomials, let $f\colon p \to q$ and $g\colon q \to r$ be natural transformations, and let $h \coloneqq f \fatsemi g$ be their composite, whose X-component $h_X\colon p(X) \to r(X)$ for each $X \in \mathbf{Set}$ is the composite of the X-components of f and g; that is, $h_X = f_X \fatsemi g_X$.

Use Corollary 2.58 to show that the arena morphism $(h_1, h^\sharp)$ associated to h satisfies $h_1 = f_1 \fatsemi g_1$ and $h_i^\sharp = g_{f_1(i)}^\sharp \fatsemi f_i^\sharp$ for all $i \in p(1)$. ◊

Example 2.61 (Commutative diagrams in **Poly**). The above exercise tells us how to interpret commutative diagrams in **Poly** as commutative diagrams in the more familiar setting of **Set**. Given polynomials p, q, r and natural transformations $f\colon p \to q, g\colon q \to r$, and $h\colon p \to r$, the diagram

$$\begin{tikzcd}
p \arrow[r, "f"] \arrow[dr, "h"'] & q \arrow[d, "g"] \\
& r
\end{tikzcd}$$

commutes in **Poly** if and only if the forwards on-positions diagram

$$\begin{tikzcd}
p(1) \arrow[r, "f_1"] \arrow[dr, "h_1"'] & q(1) \arrow[d, "g_1"] \\
& r(1)
\end{tikzcd}$$

commutes in **Set** and, for each $i \in p(1)$, the backwards on-directions diagram

$$\begin{tikzcd}
p[i] & q[f_1(i)] \arrow[l, "f_i^\sharp"'] \\
& r[h_1(i)] \arrow[ul, "h_i^\sharp"] \arrow[u, "g^\sharp_{f_1(i)}"']
\end{tikzcd}$$

commutes in **Set**. We can use this fact to determine whether a given diagram in **Poly** commutes.

Exercise 2.62 (Solution here). Verify that, for $p, q \in$ **Poly**, the polynomial $p + q$ given by the binary sum of p and q satisfies the universal property of the coproduct of p and q. That is, provide morphisms $\iota\colon p \to p + q$ and $\kappa\colon q \to p + q$, then show that for any other polynomial r with morphisms $f\colon p \to r$ and $g\colon q \to r$, there exists a unique morphism $h\colon p + q \to r$—shown dashed—making the following diagram commute:

$$\begin{tikzcd}
p \arrow[r, "\iota"] \arrow[dr, "f"'] & p+q \arrow[d, dashed, "h"] & q \arrow[l, "\kappa"'] \arrow[dl, "g"] \\
& r &
\end{tikzcd} \tag{2.63}$$

Hint: Use Example 2.61 to determine whether a diagram commutes. ◊

Exercise 2.64 (A functor **Top** → **Poly**; solution here). This exercise is for those who know what topological spaces and continuous maps are. 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? ◊

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, decisions, and data. 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.

2.4.1 The categorical product

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 products right now. These will be useful for letting many different interfaces control the same internal dynamics.

Proposition 2.65. The category **Poly** has arbitrary products.

Proof. The proof is very similar to that of Proposition 2.38.

By Proposition 2.27, the category $\mathbf{Set}^{\mathbf{Set}}$ has arbitrary products given by $\prod_{i\in I}$. The full subcategory inclusion $\mathbf{Poly}\to\mathbf{Set}^{\mathbf{Set}}$ reflects these products. It remains to show that **Poly** is closed under the operation $\prod_{i\in I}$.

Indeed, given polynomials $\{p_i\}_{i\in I}$, by (2.22), their product is

$$\prod_{i\in I} p_i \cong \prod_{i\in I}\sum_{j\in p_i(1)} y^{p_i[j]} \cong \sum_{\bar{j}\in\prod_{i\in I}p_i(1)}\prod_{i\in I} y^{p_i[\bar{j}(i)]} \cong \sum_{\bar{j}\in\prod_{i\in I}p_i(1)} y^{\sum_{i\in I}p_i[\bar{j}(i)]}, \tag{2.66}$$

which, as a coproduct of representables, is in **Poly**. □

Exercise 2.67 (Solution here).

Use (2.42) to verify that

$$\mathbf{Poly}\left(q, \prod_{i\in I} p_i\right) \cong \prod_{i\in I}\mathbf{Poly}(q, p_i)$$

for all polynomials $\{p_i\}_{i\in I}$ and q, as expected from the universal property of products. ◊

Exercise 2.68 (Solution here). Let $p_1 := y+1$, $p_2 := y+2$, and $p_3 := y^2$. What is $\prod_{i\in 3} p_i$ according to (2.66)? Is the answer what you would expect? ◊

It follows from (2.66) that the terminal object of **Poly** is 1, and that binary products are given by

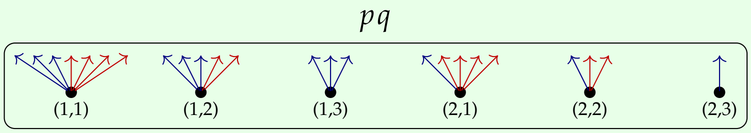
$$p \times q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i]+q[j]}. \tag{2.69}$$

We will sometimes write pq rather than $p \times q$:

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

Example 2.70. We can draw the product of two polynomials in terms of their associated forests. Let $p \coloneqq y^3 + y$ and $q \coloneqq y^4 + y^2 + 1$.

Then $pq \cong y^7 + 2y^5 + 2y^3 + y$. As arenas, we take all pairs of positions, and for each pair we take the disjoint union of the directions.



In practice, we can multiply polynomial functors the same way we would multiply two polynomials in high school algebra.

Exercise 2.71 (Solution here).

1. Show that for sets A_1, B_1, A_2, B_2, we have
$$A_1 y^{B_1} \times A_2 y^{B_2} \cong A_1 A_2 y^{B_1+B_2}.$$
2. Show that for sets $\{A_i\}_{i \in I}$ and $\{B_j\}_{j \in J}$, we have
$$\left(\sum_{i \in I} A_i y^{B_i}\right) \times \left(\sum_{j \in J} A_j y^{B_j}\right) \cong \sum_{i \in I} \sum_{j \in J} A_i A_j y^{B_i + B_j}.$$

◊

As arena morphisms, the canonical projections $\pi\colon pq \to p$ and $\phi\colon pq \to q$ behave as you might expect: on positions, they are the projections from $(pq)(1) \cong p(1) \times q(1)$ to $p(1)$ and $q(1)$, respectively; on directions, they are the inclusions $p[i] \to p[i] + q[j]$ and $q[j] \to p[i] + q[j]$ for each position (i, j) of pq.

Exercise 2.72 (Solution here). Verify that, for $p, q \in \mathbf{Poly}$, the polynomial pq given by (2.69) along with the maps $\pi\colon pq \to p$ and $\phi\colon pq \to q$ described above satisfy the universal property of the product of p and q. ◊

2.4.2 The parallel product

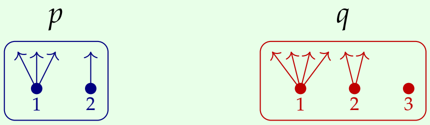
There is a closely related monoidal structure on **Poly** that will be useful for putting dynamical systems in parallel and then wiring them together.

Definition 2.73. 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:

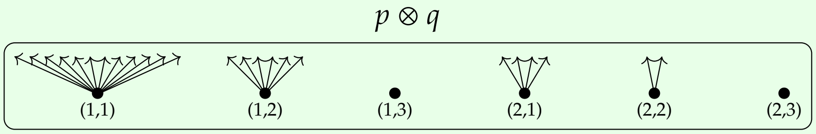
$$p \otimes q \cong \sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i] \times q[j]}. \tag{2.74}$$

One should compare this with the formula for the product of polynomials shown in (2.69). The difference is that the parallel product multiplies exponents where the categorical product adds them.

Example 2.75. We can draw the parallel product of two polynomials in terms of their associated forests. Let $p := y^3 + y$ and $q := y^4 + y^2 + 1$.



Then $p \otimes q \cong y^{12} + y^6 + y^4 + y^2 + 2$. As arenas, we take all pairs of positions, and for each pair we take the product of the directions.



Exercise 2.76 (Solution here).

1. Show that for sets A_1, B_1, A_2, B_2, we have

$$A_1 y^{B_1} \otimes A_2 y^{B_2} \cong A_1 A_2 y^{B_1 B_2}.$$

2. Show that for sets $\{A_i\}_{i \in I}$ and $\{B_j\}_{j \in J}$, we have

$$\left(\sum_{i \in I} A_i y^{B_i}\right) \otimes \left(\sum_{j \in J} A_j y^{B_j}\right) \cong \sum_{i \in I} \sum_{j \in J} A_i A_j y^{B_i B_j}.$$

◊

Exercise 2.77 (Solution here). Let $p \coloneqq y^2 + y$ and $q \coloneqq 2y^4$.

1. Draw p and q as corolla forests.
2. Draw $pq = p \times q$ as a corolla forest.
3. Draw $p \otimes q$ as a corolla forest. ◊

Exercise 2.78 (Solution here). 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!*:

$$(2 \cdot 2^y + 3 \cdot 1^y) \cdot (1 \cdot 4^y + 3 \cdot 3^y)$$

   The factors of the above product are called Dirichlet series.
4. Describe the connection between the last two parts. (An alternative name we give for the parallel product $\otimes$ is the *Dirichlet product*.) ◊

Exercise 2.79 (Solution here). What is $(3y^5 + 6y^2) \otimes 4$? Hint: $4 = 4y^0$. ◊

Exercise 2.80 (Solution here). Let $p, q, r \in \mathbf{Poly}$ 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 2.80, we have gone most of the way to proving that $(\mathbf{Poly}, y, \otimes)$ is a symmetric monoidal category.

Proposition 2.81. The category **Poly** has a symmetric monoidal structure $(y, \otimes)$ where $\otimes$ is the parallel product from Definition 2.73.

Sketch of proof. Given maps of polynomials $f \colon p \to p'$ and $g \colon q \to q'$, we need to give a map $(f \otimes g) \colon (p \otimes q) \to (p' \otimes q')$. 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

$$(f \otimes g)^\sharp_{(i,j)}(d, e) \coloneqq \big(f^\sharp_i(d), g^\sharp_j(e)\big).$$

Then Exercise 2.80 gives us the unitors, associator, and braiding. We have not proven the functoriality of $\otimes$, the naturality of the isomorphisms from Exercise 2.80, or all the coherences between these isomorphisms, but we ask the reader to take them on trust or to check them for themselves. Alternatively, we may invoke the Day convolution to obtain the monoidal structure $(y, \otimes)$ directly (see Proposition 2.86). □

Exercise 2.82 (Solution here).

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 2.83 (Solution here). 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 $\{p \in \mathbf{Poly} \mid p(1) \text{ is finite}\}$. ◊

Exercise 2.84 (Solution here). What is the smallest class of polynomials that's closed under $\otimes$ and contains y? ◊

Exercise 2.85 (Solution here). Show that for any $p_1, p_2, q \in \mathbf{Poly}$ there is an isomorphism

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

Proposition 2.86. For any monoidal structure $(I, \star)$ on **Set**, there is a corresponding monoidal structure $(y^I, \odot)$ on **Poly**, where $\odot$ is the Day convolution. Moreover, $\odot$ distributes over coproducts.

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

Proof. Any monoidal structure $(I, \odot)$ on **Set** induces a monoidal structure on $\mathbf{Set}^{\mathbf{Set}}$ with the Day convolution $\odot$ as the tensor product and y^I as the unit. To prove that this monoidal structure restricts to **Poly**, it suffices to show that **Poly** is closed under the Day convolution.

Given polynomials $p \coloneqq \sum_{i \in p(1)} y^{p[i]}$ and $q \coloneqq \sum_{j \in q(1)} y^{q[j]}$, their Day convolution is given by the coend

$$p \odot q \cong \int^{(A,B) \in \mathbf{Set}^2} y^{A \star B} \times p(A) \times q(B). \tag{2.87}$$

We can rewrite the product $p(A) \times q(B)$ as

$$p(A) \times q(B) \cong \left(\sum_{i \in p(1)} A^{p[i]} \right) \times \left(\sum_{j \in q(1)} B^{q[i]} \right) \cong \sum_{(i,j) \in p(1) \times q(1)} A^{p[i]} \times B^{q[i]}$$

So because products distribute over coproducts in **Set** and coends commute with coproducts, we can rewrite (2.87) as

$$p \odot q \cong \sum_{(i,j) \in p(1) \times q(1)} \int^{(A,B) \in \mathbf{Set}^2} y^{A \star B} \times A^{p[i]} \times B^{q[i]},$$

which, by the co-Yoneda lemma, can be rewritten as

$$p \odot q \cong \sum_{(i,j) \in p(1) \times q(1)} y^{p[i] \star q[j]} \tag{2.88}$$

in **Poly**. That the Day convolution distributes over coproducts also follows from the fact that products distribute over coproducts in **Set** and coends commute with coproducts; or, alternatively, directly from (2.88).

We observe that (2.88) gives $(y^I, \odot) = (1, \times)$ when $(I, \star) = (0, +)$ and $(y^I, \odot) = (y, \otimes)$ when $(I, \star) = (1, \times)$. □

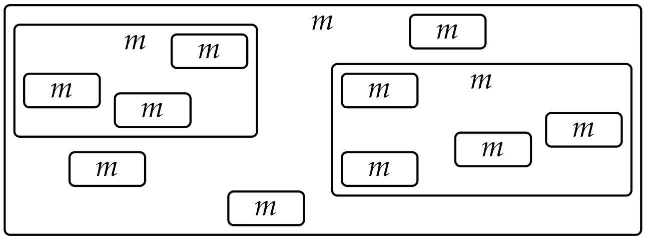
Exercise 2.89 (Solution here).

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 structure on **Poly**. Compute the monoidal product $(y^3 + y) \odot (2y^2 + 2)$. ◊

2.4.3 ⊗-monoids in Poly

Definition 2.90. A *⊗-monoid in* **Poly** consists of a polynomial m, called the *carrier*; a position $\eta \colon y \to m$, called the *unit*; and a map $\mu \colon m \otimes m \to m$, called the *multiplication*, that form a monoid in $(\mathbf{Poly}, y, \otimes)$—that is, they must satisfy certain equations. We sometimes say that *m carries a ⊗-monoid structure*, and we refer to the unit and multiplication together as a *⊗-monoid structure on m*.

These $\otimes$-monoids have a kind of swarm-like semantics, letting supervisors summarize the positions of a crew of subordinates and distribute instructions to those subordinates in a coherent way.



(2.91)

Each box in this picture is labeled m. Whenever a box has n subboxes, it represents the canonical map

$$m^{\otimes n} \to m$$

given by the monoid structure on m. We can think of this map as an (associative, unital) n-ary multiplication operation on m. Formally we might denote it by μ^{n-1}, so that $\mu^1 = \mu\colon m \otimes m \to m$,[2] but for typographical reasons it is nicer to just write μ for all of them.

The on-positions function $\mu_1\colon m(1)^n \to m(1)$ sends every n-tuple $i \coloneqq (i_1, \ldots, i_n)$ of positions to a single position $i' \coloneqq \mu_1(i)$. In the language of decision-making, we can think of this as a crew of n subordinates, each with a decision to make, summarizing all of their decisions as a single decision for their supervisor to make.

Meanwhile, the on-directions function

$$\mu_i^\sharp\colon m[i'] \to m[i_1] \times \cdots \times m[i_n]$$

at the n-tuple i sends every direction at i' to a direction at each of the n positions in i. Continuing our decision-making analogy, the supervisor chooses a single option to distribute to all the subordinates that tells them which option to pick for each of their individual decisions. This relationship between subordinates and their supervisor gives us our swarm-like semantics for $\otimes$-monoids.

The coherence of the monoid ensures that if we have a nested hierarchy in which intermediary supervisors of smaller crews are themselves subordinates of a higher supervisor, we may reassociate some of the crews or even forget about some of the intermediary supervisors without altering the relationship. In other words, we may safely ignore the intermediary subboxes in (2.91).

Example 2.92 (Monoids in **Set** give linear $\otimes$-monoids). If $(M, e, *)$ is a monoid in $(\mathbf{Set}, 1, \times)$, then the linear polynomial My carries a corresponding $\otimes$-monoid structure. Here the unit $\eta\colon y \to My$ specifies the position $e \in M$, and since $My \otimes My \cong (M \times M)y$, the multiplication is a map $\mu\colon (M \times M)y \to My$. This map is equal to $*\colon M \times M \to M$

---

[2] Here $\mu^{-1} = \eta$ and $\mu^n = (\mu^{n-1} \otimes \mathrm{id}_m) \fatsemi \mu$ for $n \in \mathbb{N}$, so that $\mu^0 = \mathrm{id}_m$ and $\mu^1 = \mu$.

on positions and the identity on directions.

We may interpret the swarm-like semantics of this $\otimes$-monoid. Every decision has but a single option, so no choices need to be made. So we can think of the decisions simply as straightforward actions; any crew of subordinates can send their actions to their supervisor as a one-action summary via $*$, and there is no need for the supervisor to send any directions back.

This construction is functorial and fully faithful: a map between monoids in **Set** can be identified with a map between $\otimes$-monoids.

Example 2.93 (Sets give representable $\otimes$-monoids). For any set S, the polynomial y^S carries a canonical $\otimes$-monoid structure: the unit is the unique map $\eta\colon y \to y^S$, while the multiplication $\mu\colon y^S \otimes y^S \to y^S$ is the identity on positions and the diagonal $S \to S \times S$ on directions.

We may interpret the swarm-like semantics of this $\otimes$-monoid. There is only one kind of decision to be made, always with the same options. So there is no need for the subordinates to tell their supervisor what decision they are making; all the supervisor has to do is select an option, and every subordinate will know to select that option, too.

This construction is again (contravariantly) functorial and fully faithful.

Example 2.94. Let $m := \mathbb{R}_{\geq 0}y^{\mathbb{R}_{\geq 0}}$. If we take $\eta := 0$ and take $\mu_1(a,b) := a + b$ and $\mu^\sharp_{(a,b)}(t) := \left(\frac{at}{a+b}, \frac{bt}{a+b}\right)$, then (m, η, μ) is a $\otimes$-monoid.

Exercise 2.95 (Solution here). Give an interpretation of the swarm-like semantics of the $\otimes$-monoid in Example 2.94. ◊

Example 2.96. Here are two examples of $\otimes$-monoid structures carried by

$$\mathsf{List} := \sum_{n \in \mathbb{N}} y^{\mathsf{n}} = 1 + y + y^2 + \cdots,$$

the "list" polynomial, whose position-set is $\mathbb{N}$ and whose direction-set at $n \in \mathbb{N}$ is n.[a] The first $\otimes$-monoid we'll discuss is commutative, while the second $\otimes$-monoid is not. We will check that they satisfy the necessary conditions of an $\otimes$-monoid in Exercise 2.97.

We have one $\otimes$-monoid carried by ℓ that is commutative, as follows: its unit specifies the position $1 \in \mathbb{N}$, and its multiplication is a map $\ell \otimes \ell \to \ell$ whose on-positions function $\cdot\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is standard integer multplication (e.g. $2 \cdot 3 = 6$) and whose on-directions function $\mathsf{mn} \to \mathsf{m} \times \mathsf{n}$ for each $(m, n) \in \mathbb{N}$ is the canonical isomorphism between the set $\{1, \ldots, mn\}$ and the set $\{1, \ldots, m\} \times \{1, \ldots, n\}$.

We have another ⊗-monoid carried by ℓ that is not commutative, as follows: its unit specifies the position $0 \in \mathbb{N}$, and its multiplication is a map $\ell \otimes \ell \to \ell$ whose on-positions function $+\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is standard integer addition and whose on-directions function $\mathsf{m} + \mathsf{n} \to \mathsf{m} \times \mathsf{n}$ is given by

$$d \mapsto \big(\min(d, m), \max(0, d - m)\big)$$

for each $(m, n) \in \mathbb{N}$. For example, take the position $(m, n) := (3, 5)$; it is sent to position 8. Given a direction $d \in \mathsf{8}$ there, it is distributed to a pair (d_1, d_2) of directions $d_1 \in \mathsf{3}$ and $d_2 \in \mathsf{5}$. If $1 \leq d \leq 3$, then $d_1 = d$ and $d_2 = 0$; if $3 \leq d \leq 8$, then $d_1 = 3$ and $d_2 = d - 5$.

---

[a] We call this the "list" polynomial because, when viewed as a functor **Set** → **Set**, it sends each set A to the set of finite lists (i.e. sequences) of elements in A, often denoted $\mathsf{List}(A)$ or A^*. You may know this polynomial as the list monad, the free monoid (on a set), or the Kleene star.

Exercise 2.97 (Solution here).

1. Check that the first ⊗-monoid described in Example 2.96 really satisfies the necessary equations.
2. Show that the first ⊗-monoid in Example 2.96 really is commutative.
3. Give an interpretation of the swarm-like semantics of the first ⊗-monoid in Example 2.96.
4. Check that the second ⊗-monoid described in Example 2.96 really satisfies the necessary equations.
5. Show that the second ⊗-monoid in Example 2.96 really is noncommutative.
6. Give an interpretation of the swarm-like semantics of the second ⊗-monoid in Example 2.96. ◊

Exercise 2.98 (Solution here). Find a ⊗-monoid carried by the polynomial $y + 1$. ◊

Proposition 2.99. If m, n are the carriers of ⊗-monoids, then $m \otimes n$ naturally also carries a ⊗-monoid structure.

Exercise 2.100 (Solution here). Sketch a proof of Proposition 2.99. ◊

Exercise 2.101 (Solution here). Give a ⊗-monoid structure on $\mathbb{N}y^5$. ◊

Proposition 2.102 (Free ⊗-monoid)**.** The forgetful functor from ⊗-monoids to **Poly** has a left adjoint sending p to a ⊗-monoid whose carrier is $\sum_{n \in \mathbb{N}} p^{\otimes n}$.

Proof. By Proposition 2.86, $\otimes$ distributes over coproducts, so the result follows from [Mac98, Chapter VII, Theorem 2] (or [nLa21, Proposition 2.2]). In particular, the unit of the free $\otimes$-monoid is the unique position of the summand $p^{\otimes 0} \cong y$ of $\sum_{n \in \mathbb{N}} p^{\otimes n}$, while the multiplication has as its components the canonical inclusions

$$p^{\otimes i} \otimes p^{\otimes j} \cong p^{\otimes(i+j)} \to \sum_{n \in \mathbb{N}} p^{\otimes n}$$

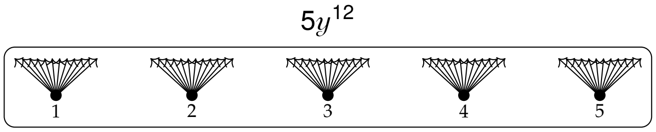
for $i, j \in \mathbb{N}$. □

Proposition 2.102 tells us that the carrier of the free $\otimes$-monoid on p has as its positions elements of $\mathsf{List}(p(1))$, i.e. lists of positions of p. Given a position $\ell = (i_1, \ldots, i_n)$, the set of directions there is given by $\prod_{j \in \mathsf{n}} p[i_j]$; i.e., the directions of the carrier at a list of positions of p are just directions of p at each position in the list.

Exercise 2.103 (Solution here). Give an interpretation of the swarm-like semantics of the free $\otimes$-monoid on a polynomial $p \in \mathbf{Poly}$. ◊

2.4.4 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:

$$\begin{aligned} \mathbf{Poly}\left(A_1 y^{B_1}, A_2 y^{B_2}\right) &\cong \prod_{a \in A_1} \sum_{a' \in A_2} B_1^{B_2} \\ &\cong \mathbf{Set}(A_1, A_2 \times B_1^{B_2}) \\ &\cong \mathbf{Set}(A_1, A_2) \times \mathbf{Set}(A_1 \times B_2, B_1). \end{aligned} \tag{2.42}$$

It says that to give a morphism from one monomial to another, you just need to give two (non-dependent!) functions. Let's rewrite it to make those two functions explicit—they are the familiar on-positions and on-directions functions:

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

Ordinarily, $f^\sharp$ is more involved: its type depends on the directions at each input position and output position of f_1. But for monomials, every position has the same set of directions, so $f^\sharp$ is just a standard function.

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* [Hed18]. It comes up in functional programming. The functions $f_1, f^\sharp$ corresponding to a morphism $f\colon A_1y^{B_1} \to A_2y^{B_2}$ are given special names:

$$\begin{aligned} \text{get} &\coloneqq f_1\colon A_1 \to A_2 \\ \text{set} &\coloneqq f^\sharp\colon A_1 \times B_2 \to B_1 \end{aligned} \tag{2.104}$$

The idea is that each position $a \in A_1$ of $A_1y^{B_1}$ "gets" a position $f_1(a) \in A_2$ of $A_2y^{B_2}$, and given $a \in A_1$, every direction at $f_1(a)$ in B_2 "sets" a direction back at a in B_1.

We will call a polynomial morphism between two monomials a *lens* when we want to remind the reader that morphisms between monomials are much simpler than those between arbitrary polynomials.

Remark 2.105. Consider the monomial Sy^S. As an arena, the set of positions is S, and the set of directions at each position $s \in S$ is again just S. In the language of decision-making, each $s \in S$ is a decision where the options you have to choose from are always just the decisions in S again. Notice that there is a natural way to string together a series of such decisions into a cycle: at each step, you start at some element of S, and the option you select is the element of S that you will move to next. We will formalize this idea in **??**.

A lens (i.e. a polynomial map) (get, set): $Sy^S \to Ty^T$ is as usual a way to delegate decisions of Sy^S to decisions of Ty^T. When you need to make a decision at $s \in S$, you ask your friend at $\text{get}(s) \in T$ for help. If your friend selects option $t \in T$, then you know to select option $\text{set}(s, t) \in S$.

But what happens when we string together these decisions into cycles? Now you are moving between elements of S, looking to your friend for help at each step as they move between elements of T. In this scenario, there are a few conditions that a lens $Sy^S \to Ty^T$ should satisfy to ensure that the associated delgation behaves well with respect to the movements of both you and your friend:

1. If your friend chooses to stay put, then you should stay put, too. This is reflected by the equation
$$\text{set}(s, \text{get}(s)) = s.$$
2. After your friend moves, and you move accordingly, you should delegate the decision at your new location to the decision at your friend's new location. This is reflected by the equation
$$\text{get}(\text{set}(s, t)) = t.$$
3. If your friend moves to t, then to t', the place where you end up should be where you would have ended up if your friend had moved directly to t' in the first place. This is reflected by the equation
$$\text{set}(\text{set}(s, t), t') = \text{set}(s, t')$$

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

2.5 Summary and further reading

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

2.6 Exercise solutions

Solution to Exercise 2.4.

Our goal is to say whether various functors are representable (of the form $X \mapsto \mathbf{Set}(S, X)$ for some S, called the representing set).

1. The identity functor $X \mapsto X$ is represented by $S = 1$: a function $1 \to X$ is just an element of X, so $\mathbf{Set}(1, X) \cong X$. Alternatively, note that $X^1 \cong X$.
2. The constant functor $X \mapsto 2$ is not representable: the functor sends 1 to 2, but $1^S \cong 1 \not\cong 2$ for any set S.
3. The constant functor $X \mapsto 1$ is representable by $S = 0$: there is exactly one function $0 \to X$, so $\mathbf{Set}(0, X) \cong 1$. Alternatively, note that $X^0 \cong 1$.
4. The constant functor $X \mapsto 0$ is not representable, for the same reason as in #2.
5. The functor $y^{\mathbb{N}}$ that sends $X \mapsto X^{\mathbb{N}}$ is represented by $S = \mathbb{N}$, by definition. It sends each function $h\colon X \to Y$ to the function $h^{\mathbb{N}}\colon X^{\mathbb{N}} \to Y^{\mathbb{N}}$ that sends each $g\colon \mathbb{N} \to X$ to $g \mathbin{⨟} h\colon \mathbb{N} \to Y$.
6. No $\mathbf{Set} \to \mathbf{Set}$ functor $X \mapsto 2^X$ is representable, for the same reason as in #2. (There *is*, however, a functor $\mathbf{Set}^{\text{op}} \to \mathbf{Set}$ sending $X \mapsto 2^X$ that is understood to be representable in a more general sense.)

Solution to Exercise 2.6.

To show that

$$\begin{tikzcd}
X^S \arrow[r, "h^S"] \arrow[d, "X^f"'] \arrow[dr, phantom, "?"] & Y^S \arrow[d, "Y^f"] \\
X^R \arrow[r, "h^R"'] & Y^R
\end{tikzcd}$$

commutes, we note that by Proposition 2.5, both vertical maps compose functions from S with $f\colon R \to S$ from the left, and by Definition 2.1, both horizontal maps compose functions to X with $h\colon X \to Y$ on the right. So by the associativity of composition, the diagram commutes.

Solution to Exercise 2.7.

In each case, given $f\colon R \to S$, we can find the X-component $X^f\colon X^S \to X^R$ of the natural transformation $y^f\colon y^S \to y^R$ by applying Proposition 2.5, which says that X^f sends each $g\colon S \to X$ to $f \mathbin{⨟} g\colon R \to X$.

1. If $R = 5$, $S = 5$, and $f = \text{id}$, then X^f is the identity function on X^5.
2. If $R = 2$, $S = 1$, and f is the unique function, then X^f sends each $g \in X$ (i.e. function $g\colon 1 \to X$) to the function that maps both elements of 2 to g. We can think of X^f as the diagonal $X \to X \times X$.
3. If $R = 1$, $S = 2$, and $f(1) = 1$, then X^f sends each $g\colon 2 \to X$ to $g(1)$, viewed as a function $1 \to X$. We can think of X^f as the left projection $X \times X \to X$.
4. If $R = 1$, $S = 2$, and $f(1) = 2$, then X^f sends each $g\colon 2 \to X$ to $g(2)$, viewed as a function $1 \to X$. We can think of X^f as the right projection $X \times X \to X$.
5. If $R = 0$, $S = 5$, and f is the unique function, then X^f is the unique function $X^5 \to X^0 \cong 1$.
6. If $R = \mathbb{N}$, $S = \mathbb{N}$, and $f(n) = n + 1$, then X^f sends each $g\colon \mathbb{N} \to X$ to the function $h\colon \mathbb{N} \to X$ satisfying $h(n) = g(n + 1)$ for all $n \in \mathbb{N}$. We can think of X^f as removing the first term of an infinite sequence of elements of X.

Solution to Exercise 2.8.

1. The fact that $y^{\mathrm{id}_S}\colon y^S \to y^S$ is the identity is just a generalization of Exercise 2.7 #1. For any set X, the X-component $X^{\mathrm{id}_S}\colon X^S \to X^S$ of y^{id_S} sends each $h\colon S \to X$ to $\mathrm{id}_S \fatsemi h = h$, so X^{id_S} is the identity on X^S. Hence y^{id_S} is the identity on y^S.
2. Fix $f\colon R \to S$ and $g\colon S \to T$; we wish to show that $y^g \fatsemi y^f = y^{f \fatsemi g}$. It suffices to show componentwise that $X^g \fatsemi X^f = X^{f \fatsemi g}$ for every set X. Indeed, X^g sends each $h\colon T \to X$ to $g \fatsemi h$; then X^f sends $g \fatsemi h$ to $f \fatsemi g \fatsemi h = X^{f \fatsemi g}(h)$.

Solution to Exercise 2.12.

1. Given $a \in F(S)$, naturality of the maps $X^S \to F(X)$ that send $g\colon S \to X$ to $F(g)(a)$ amounts to the commutativity of

$$\begin{tikzcd}
X^S \arrow[r, "h^S"] \arrow[d, "F(-)(a)"'] & Y^S \arrow[d, "F(-)(a)"] \\
F(X) \arrow[r, "F(h)"'] & F(Y)
\end{tikzcd}$$

for all $h\colon X \to Y$. The top map h^S sends any $g\colon X \to S$ to $g \fatsemi h$ (Definition 2.1), which is then sent to $F(g \fatsemi h)(a)$ by the right map. Meanwhile, the left map sends g to $F(g)(a)$, which is then sent to $F(h)(F(g)(a))$ by the bottom map. So by the functoriality of F, the square commutes.

2. We seek to show that the two maps from the proof sketch of Lemma 2.10 are mutually inverse. First, we show that for any natural transformation $m\colon y^S \to F$, we have that $m^{m_S(\mathrm{id}_S)} = m$. Given a set X, the X-component of $m^{m_S(\mathrm{id}_S)}$ sends each $g\colon S \to X$ to $F(g)(m_S(\mathrm{id}_S))$; it suffices to show that this is also where the X-component of m sends g. Indeed, by the naturality of m, the square

$$\begin{tikzcd}
S^S \arrow[r, "g^S"] \arrow[d, "m_S"'] & X^S \arrow[d, "m_X"] \\
F(S) \arrow[r, "F(g)"'] & F(X)
\end{tikzcd}$$

commutes, so in particular

$$F(g)(m_S(\mathrm{id}_S)) = m_X(g^S(\mathrm{id}_S)) = m_X(\mathrm{id}_S \fatsemi g) = m_X(g). \tag{2.106}$$

In the other direction, we show that for any $a \in F(S)$, we have $m^a_S(\mathrm{id}_S) = a$: by construction, $m^a_S\colon S^S \to F(S)$ sends id_S to $F(\mathrm{id}_S)(a) = a$.

3. Given functors $F, G\colon \mathbf{Set}^{\mathbf{Set}}$ and a natural transformation $\alpha\colon F \to G$, we wish to show that the naturality square

$$\begin{tikzcd}
\mathsf{Nat}(y^S, F) \arrow[r, "\sim"] \arrow[d, "- \fatsemi \alpha"'] & F(S) \arrow[d, "\alpha_S"] \\
\mathsf{Nat}(y^S, G) \arrow[r, "\sim"] & G(S)
\end{tikzcd}$$

commutes. The top map sends any $m\colon y^S \to F$ to $m_S(\mathrm{id}_S)$, which in turn is sent by the right map to $\alpha_S(m_S(\mathrm{id}_S)) = (m \fatsemi \alpha)_S(\mathrm{id}_S)$. This is also where the bottom map sends $m \fatsemi \alpha$, so the square commutes.

4. Given a function $g\colon S \to X$, we wish to show that the naturality square on the left side of the diagram

$$\begin{tikzcd}
\mathsf{Nat}(y^S, F) \arrow[r, "\sim"] \arrow[d, "y^g \fatsemi -"'] & F(S) \arrow[d, "F(f)"] \\
\mathsf{Nat}(y^X, F) \arrow[r, "\sim"] & F(X)
\end{tikzcd}$$

commutes. The left map sends any $m\colon y^S \to F$ to $y^g \fatsemi m$, which is sent by the bottom map to $(y^g \fatsemi m)_X(\mathrm{id}_X) = m_X(X^g(\mathrm{id}_X)) = m_X(f \fatsemi \mathrm{id}_X) = m_X(g)$. Meanwhile, the top map sends m to $m_S(\mathrm{id}_S)$, which is sent by the right map to $F(g)(m_S(\mathrm{id}_S))$. So the square commutes by (2.106).

5. To show that $\mathsf{Nat}(y^S, y^T) \cong S^T$, just take $F = y^T$ in Lemma 2.10.

Solution to Exercise 2.16.

We are given a set I and a dependent set $(X_i)_{i \in I}$ for which $X_i \coloneqq \mathsf{1}$ for every $i \in I$.

1. To show that $I \cong \sum_{i \in I} \mathsf{1}$, we note that $x \in \mathsf{1} = \{1\}$ if and only if $x = 1$, so $\sum_{i \in I} \mathsf{1} = \{(i, 1) \mid i \in I\}$. Then function $I \to \sum_{i \in I} \mathsf{1}$ that sends each $i \in I$ to $(i, 1)$ is clearly an isomorphism.
2. To show that $\mathsf{1} \cong \prod_{i \in I} \mathsf{1}$, it suffices to demonstrate that there is a unique dependent function $f \colon (i \in I) \to \mathsf{1}$. As $\mathsf{1} = \{1\}$, such a function f must always send $i \in I$ to 1. This completely characterizes f, so there is only one such dependent function.

Now $I \coloneqq \varnothing$ and $X \colon \varnothing \to \mathbf{Set}$ is the unique empty collection of sets.

3. Yes: since I is empty, there are no $i \in I$. So it is true that $X_i = 1$ holds whenever $i \in I$ holds, because $i \in I$ never holds. We say that this sort of statement is "vacuously true."
4. As $I = \varnothing = \mathsf{0}$, we have $\mathsf{0} = I \cong \sum_{i \in I} \mathsf{1} = \sum_{i \in \varnothing} X_i$, where the middle isomorphism follows from #1 and the last equation follows from #3.
5. As $I = \varnothing = \mathsf{0}$, we have $\mathsf{1} \cong \prod_{i \in I} \mathsf{1} = \prod_{i \in \varnothing} X_i$, where the isomorphism on the left follows from #2 and the equation on the right follows from #3.

Solution to Exercise 2.17.

We have a dependent set $X \colon I \to \mathbf{Set}$ and a projection function $\pi_1 \colon \sum_{i \in I} X_i \to I$ defined by $\pi_1(i, x) = i$.

1. The second projection $\pi_2(i, x) = x$ sends each pair $p = (i, x) \in \sum_{i \in I} X_i$ to x, an element of X_i. Note that we can write i in terms of p as $\pi_1(p)$. This allows us to write the signature of π_2 as $\pi_2 \colon (p \in \sum_{i \in I} X_i) \to X_{\pi_1(p)}$.
2. Let $S \coloneqq \{s \colon I \to \sum_{i \in I} X_i \mid s \mathbin{⨟} \pi_1 = \mathrm{id}_I\}$ be the set of sections of π_1. To show that $\prod_{i \in I} X_i \cong S$, we will exhibit maps in either direction and show that they are mutually inverse. For each $f \colon (i \in I) \to X_i$ in $\prod_{i \in I} X_i$, we have that $f(i) \in X_i$ for all $i \in I$, so we can define a function $s_f \colon I \to \sum_{i \in I} X_i$ that sends each $i \in I$ to $(i, f(i))$. Then $\pi_1(s_f(i)) = \pi_1(i, f(i)) = i$, so s_f is indeed a section of π_1. Hence $f \mapsto s_f$ is a map $\prod_{i \in I} X_i \to S$.
   In the other direction, for each section $s \colon I \to \sum_{i \in I} X_i$ we have $\pi_1(s(i)) = i$ for all $i \in I$, so we can write $s(i)$ as an ordered pair $(i, \pi_2(s(i)))$ with $\pi_2(s(i)) \in X_i$. It follows that we can define a function $f_s \colon (i \in I) \to X_i$ that sends each $i \in I$ to $\pi_2(s(i))$. Then $s \mapsto f_s$ is a map $S \to \prod_{i \in I} X_i$. By construction $s_{f_s}(i) = (i, f_s(i)) = (\pi_1(s(i)), \pi_2(s(i))) = s(i)$ and $f_{s_f}(i) = \pi_2(s_f(i)) = \pi_2(i, f(i)) = f(i)$, so these maps are mutually inverse.

Solution to Exercise 2.20.

We are given the set

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

1. Here are the instructions for choosing an element of B as a nested list.

```
To choose an element of $B$:
  1. for each element $i \in I$:
    1. choose an element $j \in J(i)$;
    2. for each element $k \in K(i, j)$:
      1. choose an element of $X(i, j, k)$.
```

2. Given $I \coloneqq \{1, 2\}$, $J(1) \coloneqq \{j\}$, $J(2) \coloneqq \{j, j'\}$, $K(1, j) \coloneqq \{k_1, k_2\}$, $K(2, j) \coloneqq \{k_1\}$, $K(2, j') \coloneqq \{k'\}$, and $X(i, j, k) = \{x, y\}$ for all i, j, k, our goal is to count the number of elements in B. To compute the cardinality of B, we can use the fact that the cardinality of a sum (resp. product) is the sum (resp. product) of the cardinalities. So

$$\begin{aligned} |B| &= \prod_{i \in I} \sum_{j \in J(i)} \prod_{k \in K(i,j)} |X(i, j, k)| \\ &= \prod_{i \in \{1,2\}} \sum_{j \in J(i)} \prod_{k \in K(i,j)} 2 \end{aligned}$$

$$
\begin{aligned}
&= \left(\sum_{j\in J(1)} 2^{|K(1,j)|}\right)\left(\sum_{j\in J(2)} 2^{|K(2,j)|}\right)\\
&= \left(2^2\right)\left(2^1+2^1\right) = 16.
\end{aligned}
$$

3. Here are three of the elements of B (you may have written down others):
   - $(1\mapsto(j,k_1\mapsto x,k_2\mapsto y),2\mapsto(j',k'\mapsto x))$
   - $(1\mapsto(j,k_1\mapsto y,k_2\mapsto y),2\mapsto(j,k_1\mapsto y))$
   - $(1\mapsto(j,k_1\mapsto y,k_2\mapsto x),2\mapsto(j',k'\mapsto y))$

Solution to Exercise 2.26.

1. It suffices to show that for all $X\in\mathbf{Set}$,

$$\sum_{i\in I} F(X)\cong (IF)(X).$$

   The left hand side is isomorphic to the set $\{(i,s)\mid i\in I \text{ and } s\in F(X)\}\cong I\times F(X)$, while the right hand side is also isomorphic to the set $I(X)\times F(X)\cong I\times F(X)$. (Alternatively, the result also follows from (2.22).)
2. It doesn't matter: $2y$ can denote either one. Indeed, taking $I:=2$, the above says that there is an isomorphism $2\times y\cong y+y$.

Solution to Exercise 2.31.

Our goal is to write out the elements of $p(X)$ for each polynomial p and set X that we are given.

1. If $p:=y^3$ and $X:=\{4,9\}$, then let $I:=1$ and $p[1]:=3$ so that $p\cong\sum_{i\in I}y^{p[i]}$. So

$$p(X)\cong\{(1,4,4,4),(1,4,4,9),(1,4,9,4),(1,4,9,9),(1,9,4,4),(1,9,4,9),(1,9,9,4),(1,9,9,9)\}.$$

2. If $p:=3y^2+1$ and $X:=\{a\}$, then let $I:=4$, $p[1]:=p[2]:=p[3]:=2$, and $p[4]:=1$ so that $p\cong\sum_{i\in I}y^{p[i]}$. So $p(X)\cong\{(1,a,a),(2,a,a),(3,a,a),(4)\}$.
3. If $:=0$ and $X:=\mathbb{N}$, then let $I:=0$ so that $p\cong\sum_{i\in I}y^{p[i]}$. So $p(X)\cong 0$. Alternatively, we note that 0 is the constant functor that assigns 0 to every set.
4. If $p:=4$ and $X:=\mathbb{N}$, then let $I:=4$ and $p[i]:=0$ for every $i\in I$. So $p(X)\cong\{(1),(2),(3),(4)\}\cong 4$.
5. If $p:=y$ and $X:=\mathbb{N}$, then let $I:=1$ and $p[1]:=1$ so that $p\cong\sum_{i\in I}y^{p[i]}$. So $p(X)\cong\{(1,n)\mid n\in\mathbb{N}\}$.

Solution to Exercise 2.33.

Given $p:=y^2+y$, we seek the induced function $p(f)\colon p(1)\to p(2)$ for a function $f\colon 1\to 2$ of our choice. We will choose $1\mapsto 2$. We can evaluate

$$p(1)\cong\{(1,1,1),(2,1)\} \text{ and } p(2)\cong\{(1,1,1),(1,1,2),(1,2,1),(1,2,2),(2,1),(2,2)\}.$$

So $p(f)$ sends $(1,1,1)\mapsto(1,2,2)$ and $(2,1)\mapsto(2,2)$. (If we had instead picked $1\mapsto 1$ as our function f, then $p(f)$ would send $(1,1,1)\mapsto(1,1,1)$ and $(2,1)\mapsto(2,1)$.)

Solution to Exercise 2.36.

We consider $p(0)$ for arbitrary polynomials p. A representable functor y^S for some $S\in\mathbf{Set}$ sends $0\mapsto 0$ if $S\neq 0$ (as there are then no functions $S\to 0$), but sends $0\mapsto 1$ if $S=0$ (as there is a unique function $0\to 0$). So

$$p(0)\cong\sum_{i\in p(1)}(y^{p[i]})(0)\cong\sum_{\substack{i\in p(1),\\ p[i]\neq 0}}0+\sum_{\substack{i\in p(1),\\ p[i]=0}}1\cong\{i\in p(1)\mid p[i]=0\}.$$

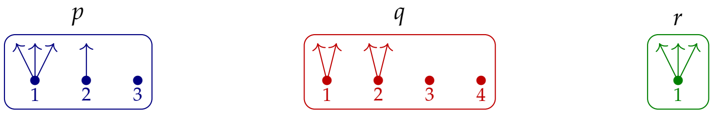
In other words, $p(0)$ is the set of *constant* representable summands of p. For example, if $p:=y^3+3y^2+4$, then $p(0)=4$. In the language of high school algebra, we might call $p(0)$ the "constant term" of p.

Solution to Exercise 2.40.

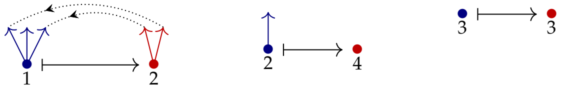
As $\mathbf{Poly}(y^S, q)$ is defined to be $\mathsf{Nat}(y^S, q)$ the natural isomorphism $\mathbf{Poly}(y^S, q) \cong q(S)$ follows directly from the Yoneda lemma (Lemma 2.10) with $F = q$.

Solution to Exercise 2.45.

1. Here are the corolla forests associated to $p \coloneqq y^3 + y + 1$, $q \coloneqq y^2 + y^2 + 2$, and $r \coloneqq y^3$ (with each root labeled for convenience).



2. Here is one possible morphism $p \to q$ (you may have drawn others).



3. As depicted, our morphism delegates the first decision of p to the second decision of q, whose first and second options are passed back to the third and first options, respectively, of the first decision of p. Then the second decision of p is delegated to the fourth decision of q, which has no options; effectively, the second decision of p has been canceled. Finally, the third decision of p is delegated to the third decision of q, neither of which has any options.
4. There cannot be a morphism $p \to r$ for the following reason: if we delegate the third decision of p, which has no options, to the sole decision of r, which has 3 options, then there is no way to pass a choice of one of the 3 options back to any of the options associated with the third decision of p, as there are no such options.

Solution to Exercise 2.46.

1. We let $p \coloneqq y^3 + 1$ (you could have selected others) and draw both p and y^2 as corolla forests, labeling each root for convenience.

2. We count all the maps from y^2 to p. The unique position of y^2 can be sent to either position of p. If it is sent to the first position of p, then there are 2 directions of y^2 for each of the 3 directions at the first position of p to be sent to, for a total of $2^3 = 8$ maps. Otherwise, the unique position of y^2 is sent to the second position of p—at which there are no directions, so there is exactly 1 way to do this. So we have $8 + 1 = 9$ maps from y^2 to p.
3. Yes, the previous answer is equal to $p(2) = 2^3 + 1 = 9$.

Solution to Exercise 2.47.

Our goal is to compute the number of natural transformations $p \to q$ for each of the following polynomials p, q. By (2.42), we always have

$$|\mathbf{Poly}(p,q)| = \prod_{i \in p(1)} |q(p[i])|.$$

1. If $p = y^3$ and $q = y^4$, then

$$|\mathbf{Poly}(p,q)| = \prod_{i \in 1} |p[i]|^4 = 3^4 = 81.$$

2. If $p = y^3 + 1$ and $q = y^4$, then
$$|\mathbf{Poly}(p,q)| = \prod_{i \in 2} |p[i]|^4 = 3^4 \cdot 0^4 = 0.$$
3. If $p = y^3 + 1$ and $q = y^4 + 1$, then
$$|\mathbf{Poly}(p,q)| = \prod_{i \in 2} |p[i]|^4 + 1 = (3^4 + 1)(0^4 + 1) = 82.$$
4. If $p = 4y^3 + 3y^2y$ and $q = y$, then
$$|\mathbf{Poly}(p,q)| = \prod_{i \in 8} |p[i]| = 3^4 \cdot 2^3 \cdot 1 = 648.$$
5. If $p = 4y^3$ and $q = 3y$, then
$$|\mathbf{Poly}(p,q)| = \prod_{i \in 4} 3|p[i]| = (3 \cdot 3)^4 = 6561.$$

Solution to Exercise 2.48.

1. We will show that, yes, the following isomorphism does hold:
$$\mathbf{Poly}(p,q) \cong \prod_{i \in p(1)} \sum_{j \in q(1)} \prod_{d \in q[j]} \sum_{c \in p[i]} 1.$$
By (2.42), it suffices to show that for all $i \in p(1)$ and $j \in q(1)$, we have
$$p[i]^{q[j]} \cong \prod_{d \in q[j]} \sum_{c \in p[i]} 1.$$
Indeed, by (2.24) and Exercise 2.16, we have
$$\begin{aligned} \prod_{d \in q[j]} \sum_{c \in p[i]} 1 &\cong \sum_{\bar{c}\colon q[j] \to p[i]} \prod_{d \in q[j]} 1 && (2.24) \\ &\cong \sum_{\bar{c}\colon q[j] \to p[i]} 1 && \text{(Exercise 2.16 \#2)} \\ &\cong \mathbf{Set}(q[j], p[i]) && \text{(Exercise 2.16 \#1)} \\ &\cong p[i]^{q[j]}. \end{aligned}$$

2. We will show that, yes, the following isomorphism does hold:
$$\mathbf{Poly}(p,q) \cong \sum_{f_1\colon p(1) \to q(1)} \prod_{j \in q(1)} \mathbf{Set}\left(q[j], \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]\right).$$
By (2.42) and (2.24), we have
$$\begin{aligned} \mathbf{Poly}(p,q) &\cong \prod_{i \in p(1)} \sum_{j \in q(1)} p[i]^{q[j]} && (2.42) \\ &\cong \sum_{f_1\colon p(1) \to q(1)} \prod_{i \in p(1)} p[i]^{q[f_1(i)]} && (2.24) \\ &\cong \sum_{f_1\colon p(1) \to q(1)} \prod_{j \in q(1)} \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]^{q[j]} && (*) \\ &\cong \sum_{f_1\colon p(1) \to q(1)} \prod_{j \in q(1)} \mathbf{Set}\left(q[j], \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]\right) && \text{(Universal property of products)} \end{aligned}$$
where $(*)$ follows from the fact that for any function $f_1\colon p(1) \to q(1)$, the set $p(1)$ can be written as the disjoint union of sets of the form $f_1^{-1}(j) = \{i \in p(1) \mid f_1(i) = j\}$ for each $j \in q(1)$.

3. To explain how the set

$$D_{p,q} \coloneqq \sum_{f_1 \colon p(1) \to q(1)} \prod_{j \in q(1)} \mathbf{Set}\left(q[j], \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]\right)$$

specifies a way of delegating decisions from p to q, we first give the instructions for choosing an element of $D_{p,q}$ as a nested list:

To choose an element of $D_{p,q}$:

1. choose a function $f_1 \colon p(1) \to q(1)$;
2. for each element $j \in q(1)$:
   1. for each element of $q[j]$:
      1. for each element $i \in p(1)$ satisfying $f_1(i) = j$:
         1. choose an element of $p[i]$.

So f_1 delegates each of p's decisions to one of q's decisions. Then for every option of every decision j of q, we choose an option of each of p's decisions that has been delegated to j by f_1.

Solution to Exercise 2.51.

Given $q \in \mathbf{Poly}$ and $p_i \in \mathbf{Poly}$ for each $i \in I$ for some set I, we use (2.39) and (2.42) to verify that

$$\begin{aligned}
\mathbf{Poly}\left(\sum_{i \in I} p_i, q\right) &\cong \mathbf{Poly}\left(\sum_{(i,j) \in \sum_{i \in I} p_i(1)} y^{p_i[j]}, q\right) && (2.39)\\
&\cong \prod_{(i,j) \in \sum_{i \in I} p_i(1)} q(p_i[j]) && (2.42)\\
&\cong \prod_{i \in I} \prod_{j \in p_i(1)} q(p_i[j])\\
&\cong \prod_{i \in I} \mathbf{Poly}(p_i, q). && (2.42)
\end{aligned}$$

Solution to Exercise 2.53.

We can evaluate $\dot{p}(1)$ directly from the definition of $\dot{p}$ to obtain

$$\dot{p}(1) = \sum_{i \in p(1)} \sum_{d \in p[i]} 1^{p[i] - \{d\}} \cong \sum_{i \in p(1)} p[i],$$

which is isomorphic to the set of all directions of p. Then $\pi_p \colon \dot{p}(1) \to p(1)$ is the canonical projection, sending each direction d of p to the position i of p for which $d \in p[i]$.

Solution to Exercise 2.54.

Here $p, q \in \mathbf{Poly}$.

1. Our goal is to characterize the canonical map $\dot{p}y \to p$. If we unravel the definitions, this is a map
$$\sum_{i \in p(1)} \sum_{d \in p[i]} y^{p[i]} \to \sum_{i \in p(1)} y^{p[i]}.$$
We observe that there is always such a map sending every position $(i, d) \in \sum_{i \in p(1)} p[i]$ of $\dot{p}y$ to its first projection $i \in p(1)$ and is the identity on directions. This is the canonical map.
2. There cannot always be a canonical map $p \to \dot{p}$, for if $p \coloneqq 1$, then $\dot{p} \coloneqq 0$, and there is no map $1 \to 0$.
3. We show that there cannot always be a canonical map $\dot{p} \to p$. Take $p \coloneqq y$, so $\dot{p} \coloneqq 1$. A map $1 \to y$ must have an on-directions function $1 \to 0$, but such a function does not exist.
4. We show that even when there is a map $p \to q$, there is not necessarily a map $\dot{p} \to \dot{q}$. Take $p \coloneqq y$ and $q \coloneqq 1$. Then there is a map $p \to q$ that sends the unique position of y to the unique position of 1 and is the empty function on directions. But $\dot{p} = 1$ and $\dot{q} = 0$, and there is no map $1 \to 0$.

5. We show that there is a canonical map $g\colon [p,y]\otimes p \to \dot{p}$, where $[p,y]\otimes p$ is given by (2.55). The on-positions function g_1 takes $f \in \prod_{i\in p(1)} p[i]$ and $i \in p(1)$ and sends the pair of them to the position of $\dot{p}$ corresponding to $i \in p(1)$ and $f(i) \in p[i]$. We then have $\dot{p}[(i,f(i))] \cong p[i]$ and $([p,y]\otimes p)[(f,i)] \cong p(1)\times p[i]$, so the on-directions function $g^\sharp_{(f,i)}$ can send each $d \in p[i]$ to $(i,d) \in p(1)\times p[i]$.
6. We wish to describe a map $py \to q$ in terms of "unassigned to directions." Observe that as an arena, py has the same positions as p but has one more direction than p does at each position. We denote this extra direction at each position $i \in p(1)$ of py by $*_i$. So a map $f\colon py \to q$ sends each position i of p to a position j of q, but every direction of q at j could either be sent back to a direction of p at i or the extra direction $*_i$. We can say that an arena morphism $f\colon py \to q$ is like an arena morphism $p \to q$, except that any of the directions of q may remain un assigned—i.e. we may have partial on-directions functions.

Solution to Exercise 2.59.

We wish to show that the arena morphism $((\mathrm{id}_p)_1, (\mathrm{id}_p)^\sharp)$ associated to the identity natural transformation id_p of a polynomial p is such that $(\mathrm{id}_p)_1$ and every $(\mathrm{id}_p)^\sharp_i$ are all identity functions. Indeed, by Corollary 2.58, for each $i \in p(1)$, we have

$$((\mathrm{id}_p)_1(i), (\mathrm{id}_p)^\sharp_i) = (\mathrm{id}_p)_{p[i]}(i, \mathrm{id}_{p[i]}) = (i, \mathrm{id}_{p[i]}),$$

as $(\mathrm{id}_p)_{p[i]}$ is the identity function on $p(p[i])$.

Solution to Exercise 2.60.

Given polynomial morphisms $f\colon p \to q$, $g\colon q \to r$, and their composite $h \coloneqq f \fatsemi g$, we wish to show that the arena morphism $(h_1, h^\sharp)$ associated to h satisfies $h_1 = f_1 \fatsemi g_1$ and $h^\sharp_i = g^\sharp_{f_1(i)} \fatsemi f^\sharp_i$ for all $i \in p(1)$. Indeed, by Corollary 2.58 and Proposition 2.56, for each $i \in p(1)$, we have

$$\begin{aligned}
(h_1(i), h^\sharp_i) &= h_{p[i]}(i, \mathrm{id}_{p[i]}) \\
&= g_{p[i]}(f_{p[i]}(i, \mathrm{id}_{p[i]})) && (h = f \fatsemi g) \\
&= g_{p[i]}(f_1(i), f^\sharp_i) && (\text{Corollary } 2.58) \\
&= (g_1(f_1(i)), g^\sharp_{f_1(i)} \fatsemi f^\sharp_i). && (\text{Proposition } 2.56)
\end{aligned}$$

Solution to Exercise 2.62.

We provide $\iota\colon p \to p+q$ and $\kappa\colon q \to p+q$ as follows. On positions, they are the canonical inclusions $\iota_1\colon p(1) \to p(1)+q(1)$ and $\kappa_1\colon q(1) \to p(1)+q(1)$; on directions, they are identities. We wish to show that, for $p, q \in$ **Poly**, the polynomial $p+q$ along with ι and κ satisfy the universal property of the coproduct. That is, we must show that for any $r \in$ **Poly** and maps $f\colon r \to p$ and $g\colon r \to q$, there exists a unique map $h\colon r \to p+q$ for which the diagram (2.63) commutes.

We apply Example 2.61. In order for (2.63) to commute, it must commute on positions—that is, the following diagram of sets must commute:

$$\begin{tikzcd}
p(1) \arrow[r, "\iota_1"] \arrow[dr, "f_1"'] & p(1)+q(1) \arrow[d, dashed, "h_1"] & q(1) \arrow[l, "\kappa_1"'] \arrow[dl, "g_1"] \\
& r(1) &
\end{tikzcd} \tag{2.107}$$

But since $p(1)+q(1)$ along with the inclusions ι_1 and κ_1 form the coproduct of $p(1)$ and $q(1)$ in **Set**, there exists a unique h_1 for which (2.107) commutes. Hence h is uniquely characterized on positions. In particular, it must send each $(1,i) \in p(1)+q(1)$ with $i \in p(1)$ to $f_1(i)$ and each $(2,j) \in p(1)+q(1)$ with $j \in q(1)$ to $g_1(j)$.

Moreover, if (2.63) is to commute on directions, then for every $i \in p(1)$ and $j \in q(1)$, the following diagrams of sets must commute:

$$\begin{tikzcd}
p[i] & (p+q)[(1,i)] \arrow[l, "\iota_i^\sharp"'] & & (p+q)[(2,j)] \arrow[r, "\kappa_j^\sharp"] & q[j] \\
& r[f_1(i)] \arrow[ul, "f_i^\sharp"] \arrow[u, dashed, "h^\sharp_{(1,i)}"'] & & r[g_1(j)] \arrow[ur, "g_j^\sharp"'] \arrow[u, dashed, "h^\sharp_{(2,j)}"] &
\end{tikzcd} \tag{2.108}$$

But $(p+q)[(1,i)] \cong p[i]$ and $\iota_i^\sharp$ is the identity, so we must have $h^\sharp_{(1,i)} = f_i^\sharp$. Similarly, $(p+q)[(2,j)] \cong q[j]$ and $\kappa_j^\sharp$ is the identity, so we must have $h^\sharp_{(2,j)} = g_j^\sharp$. Hence h is also uniquely characterized on directions, so it is unique overall. Moreover, we have shown that we can define h on positions so that (2.107) commutes, and that we can define h on directions such that the diagrams in (2.108) commute. As the commutativity of the diagrams in (2.107) and (2.108) together imply the commutativity of (2.63), it follows that there exists h for which (2.63) commutes.

Solution to Exercise 2.64.

1. Given a topological space X, we can define a polynomial p_X whose positions are the points in X and whose directions at $x \in X$ are the open neighborhoods of x. In other words,
$$p_X \coloneqq \sum_{x \in X} y^{\{U \subseteq X \mid x \in U,\, U \text{ open}\}}.$$
2. For every continuous map $f \colon X \to Y$, we give a polynomial morphism $p_f \colon p_X \to p_Y$. The on-positions function is just f, while for each position $x \in X$ of p_X, the on-directions function $(p_f)_x^\sharp \colon p_Y[f(x)] \to p_X[x]$ sends each open neighborhood U of $f(x)$ to the open neighborhood $f^{-1}(U)$ of x.
3. To show that p_X is functorial in X, it suffices to show that sending continuous maps $f \colon X \to Y$ to their induced polynomial morphisms $p_f \colon p_X \to p_Y$ preserves identities and composition. First, we show that for any topological space X, the polynomial morphism p_{id_X}, where id_X is the identity map on X, is an identity morphism. By #2, the on-positions function of p_{id_X} is id_X, and for each $x \in X$ the on-directions function $(p_f)_x^\sharp \colon p_X[x] \to p_X[x]$ sends $U \in p_X[x]$ to $(\mathrm{id}_X)^{-1}(U) = U$. Hence p_{id_X} is the identity on both positions and directions; it follows from Exercise 2.59 that p_{id_X} is an identity morphism.
   We now show that for topological spaces X, Y, and Z and continuous maps $f \colon X \to Y$ and $g \colon Y \to Z$, we have $p_f \fatsemi p_g = p_{f \fatsemi g}$. By #2 and Exercise 2.60, the on-positions function of either side is equal to $f \fatsemi g$, so it suffices to show that for all $x \in X$,
$$(p_{f \fatsemi g})_x^\sharp = (p_g)^\sharp_{f(x)} \fatsemi (p_f)_x^\sharp.$$
   Again by #2, the left hand side sends each $U \in p_Z[g(f(x))]$ to $(f \fatsemi g)^{-1}(U)$, while the right hand side sends U to $f^{-1}(g^{-1}(U))$, but by elementary set theory, these sets are equal.
4. The functor is not full. Consider the spaces $X = 2$ with the indiscrete topology (i.e. the only open sets are the empty set and X) and $Y = 2$ with the discrete topology (i.e. all subsets are open). Then $p_X \cong 2y$ and $p_Y \cong 2y^2$, so our functor induces a function from the set of continuous functions $X \to Y$ to the set of polynomial morphisms $2y \to 2y^2$. We claim that this function is not surjective: in particular, consider the polynomial morphism $h \colon 2y \to 2y^2$ that is the identity on positions (and uniquely defined on directions). Then a continuous function $f \colon X \to Y$ that our functor sends to h must also be the identity on the underlying sets of X and Y. But such a function cannot be continuous, as the preimage under f of a singleton set of Y, which is open, would be a singleton set of X, which would not be open. So our functor sends no continuous function $X \to Y$ to h, and therefore is not full.
   The functor is, however, faithful: for any spaces X and Y and continuous function $f \colon X \to Y$, we can uniquely recover f from p_f by taking the on-positions function $(p_f)_1$.

Solution to Exercise 2.67.

Given $q \in \mathbf{Poly}$ and $p_i \in \mathbf{Poly}$ for each $i \in I$ for some set I, we use (2.42) to verify that

$$\begin{aligned}
\mathbf{Poly}\left(q, \prod_{i \in I} p_i\right) &\cong \prod_{k \in q(1)} \left(\prod_{i \in I} p_i\right)(q[k]) && (2.42)\\
&\cong \prod_{k \in q(1)} \prod_{i \in I} p_i(q[k])\\
&\cong \prod_{i \in I} \prod_{k \in q(1)} p_i(q[k])\\
&\cong \prod_{i \in I} \mathbf{Poly}(q, p_i). && (2.42)
\end{aligned}$$

Solution to Exercise 2.68.

Given $p_1 \coloneqq y + 1, p_2 \coloneqq y + 2$, and $p_3 \coloneqq y^2$, we compute $\prod_{i \in 3} p_i$ via (2.66) as follows:

$$\begin{aligned}
\prod_{i \in 3} p_i &\cong \sum_{\bar{j} \in \prod_{i \in 3} p_i(1)} y^{\sum_{i \in 3} p_i[\bar{j}(i)]} && (2.66)\\
&\cong \sum_{\bar{j} \colon (i \in 3) \to p_i(1)} y^{p_1[\bar{j}(1)] + p_2[\bar{j}(2)] + p_3[\bar{j}(3)]}\\
&\cong y^{p_1[1]+p_2[1]+p_3[1]} + y^{p_1[1]+p_2[2]+p_3[1]} + y^{p_1[1]+p_2[3]+p_3[1]}\\
&+ y^{p_1[2]+p_2[1]+p_3[1]} + y^{p_1[2]+p_2[2]+p_3[1]} + y^{p_1[2]+p_2[3]+p_3[1]}\\
&\cong y^{1+1+2} + y^{1+0+2} + y^{1+0+2}\\
&+ y^{0+1+2} + y^{0+0+2} + y^{0+0+2}\\
&\cong y^4 + 3y^3 + 2y^2,
\end{aligned}$$

as we might expect from standard polynomial multiplication.

Solution to Exercise 2.71.

1. We compute the product $A_1 y^{B_1} \times A_2 y^{B_2}$ using (2.69):

$$\begin{aligned}
A_1 y^{B_1} \times A_2 y^{B_2} &\cong \left(\sum_{i \in A_1} y^{B_1}\right) \times \left(\sum_{j \in A_2} y^{B_2}\right)\\
&\cong \sum_{i \in A_1} \sum_{j \in A_2} y^{B_1 + B_2}\\
&\cong A_1 A_2 y^{B_1 + B_2}.
\end{aligned}$$

2. We expand the product $\left(\sum_{i \in I} A_i y^{B_i}\right) \times \left(\sum_{j \in J} A_j y^{B_j}\right)$ by applying (2.22), with $I_1 \coloneqq I$ and $I_2 \coloneqq J$:

$$\begin{aligned}
\left(\sum_{i \in I} A_i y^{B_i}\right) \times \left(\sum_{j \in J} A_j y^{B_j}\right) &\cong \prod_{k \in 2} \sum_{i \in I_k} A_i y^{B_i}\\
&\cong \sum_{\bar{i} \in \prod_{k \in 2} I_k} \prod_{k \in 2} A_{\bar{i}(k)} y^{B_{\bar{i}(k)}}\\
&\cong \sum_{i \in I} \sum_{j \in J} A_i y^{B_i} \times A_j y^{B_j}\\
&\cong \sum_{i \in I} \sum_{j \in J} A_i A_j y^{B_i + B_j}
\end{aligned}$$

where the last isomorphism follows from #1.

Solution to Exercise 2.72.

We wish to show that, for $p, q \in \mathbf{Poly}$, the polynomial pq along with the maps $\pi \colon pq \to p$ and $\phi \colon pq \to q$ as described in the text satisfy the universal property of the product. That is, we must show that for any $r \in \mathbf{Poly}$ and maps $f \colon r \to p$ and $g \colon r \to q$, there exists a unique map $h \colon r \to pq$ for which the following diagram commutes:

$$\begin{tikzcd} r \arrow[r, "g"] \arrow[d, "f"'] \arrow[dr, dashed, "h"] & q \\ p & pq. \arrow[l, "\pi"] \arrow[u, "\phi"'] \end{tikzcd} \tag{2.109}$$

We apply Example 2.61. In order for (2.109) to commute, it must commute on positions—that is, the following diagram of sets must commute:

$$\begin{tikzcd} r(1) \arrow[r, "g_1"] \arrow[d, "f_1"'] \arrow[dr, dashed, "h_1"] & q(1) \\ p(1) & (pq)(1). \arrow[l, "\pi_1"] \arrow[u, "\phi_1"'] \end{tikzcd} \tag{2.110}$$

But since $(pq)(1) \cong p(1) \times q(1)$ along with the projections π_1 and ϕ_1 form the product of $p(1)$ and $q(1)$ in **Set**, there exists a unique h_1 for which (2.110) commutes. Hence h is uniquely characterized on positions. In particular, it must send each $k \in r(1)$ to the pair $(f_1(k), g_1(k)) \in (pq)(1)$.

Moreover, if (2.63) is to commute on directions, then for every $k \in r(1)$, the following diagram of sets must commute:

$$\begin{tikzcd} r[k] & & q[g_1(k)] \arrow[ll, "g_k^\sharp"'] \arrow[d, "\phi^\sharp_{(f_1(k), g_1(k))}"] \\ p[f_1(k)] \arrow[u, "f_k^\sharp"] \arrow[rr, "\pi^\sharp_{(f_1(k), g_1(k))}"'] & & (pq)[(f_1(k), g_1(k))]. \arrow[llu, dashed, "h_k^\sharp"'] \end{tikzcd} \tag{2.111}$$

As $(pq)[(f_1(k), g_1(k))] \cong p[f_1(k)] + q[g_1(k)]$ along with the inclusions $\pi^\sharp_{(f_1(k), g_1(k))}$ and $\phi^\sharp_{(f_1(k), g_1(k))}$ form the coproduct of $p[f_1(k)]$ and $q[g_1(k)]$ in **Set**, there exists a unique $h_k^\sharp$ for which (2.111) commutes. Hence h is also uniquely characterized on directions, so it is unique overall. Moreover, we have shown that we can define h on positions so that (2.110) commutes, and that we can define h on directions such that (2.111) commutes. As the commutativity of (2.110) and (2.111) together imply the commutativity of (2.109), it follows that there exists h for which (2.109) commutes.

Solution to Exercise 2.76.

1. We compute the product $A_1 y^{B_1} \otimes A_2 y^{B_2}$ using (2.74):

$$\begin{aligned} A_1 y^{B_1} \otimes A_2 y^{B_2} &\cong \left(\sum_{i \in A_1} y^{B_1}\right) \otimes \left(\sum_{j \in A_2} y^{B_2}\right) \\ &\cong \sum_{i \in A_1} \sum_{j \in A_2} y^{B_1 \times B_2} \\ &\cong A_1 A_2 y^{B_1 B_2}. \end{aligned}$$

2. We expand the product $\left(\sum_{i \in I} A_i y^{B_i}\right) \otimes \left(\sum_{j \in J} A_j y^{B_j}\right)$ as follows:

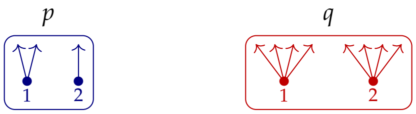
$$\begin{aligned} \left(\sum_{i \in I} A_i y^{B_i}\right) \otimes \left(\sum_{j \in J} A_j y^{B_j}\right) &\cong \left(\sum_{i \in I} \sum_{i' \in A_i} y^{B_i}\right) \otimes \left(\sum_{j \in J} \sum_{j' \in A_j} y^{B_j}\right) \\ &\cong \sum_{i \in I} \sum_{i' \in A_i} \sum_{j \in J} \sum_{j' \in A_j} y^{B_i \times B_j} \end{aligned}$$

$$\cong \sum_{i\in I}\sum_{j\in J}\sum_{i'\in A_i}\sum_{j'\in A_j} y^{B_iB_j}$$
$$\cong \sum_{i\in I}\sum_{j\in J} A_iA_jy^{B_iB_j}.$$

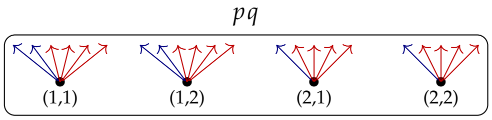
Solution to Exercise 2.77.

Here $p := y^2 + y$ and $q := 2y^4$.

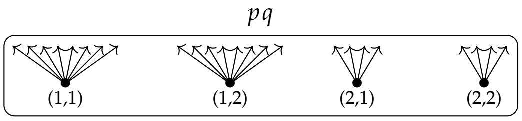
1. Here are p and q drawn as corolla forests:



2. Here is pq drawn as a corolla forest:



3. Here is $p \otimes q$ drawn as a corolla forest:



Solution to Exercise 2.78.

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

1. We compute $p \times q$ using Exercise 2.71 #2:

$$\begin{aligned} p \times q &\cong 2y^{2+4} + (2\times 3)y^{2+3} + 3y^{1+4} + (3\times 3)y^{1+3} \\ &\cong 2y^6 + 6y^5 + 3y^5 + 9y^4 \\ &\cong 2y^6 + 9y^5 + 9y^4. \end{aligned}$$

2. We compute $p \otimes q$ using Exercise 2.76 #2:

$$\begin{aligned} p \otimes q &\cong 2y^{2\times 4} + (2\times 3)y^{2\times 3} + 3y^4 + (3\times 3)y^3 \\ &\cong 2y^8 + 6y^6 + 3y^4 + 9y^3. \end{aligned}$$

3. We evaluate $(2{\cdot}2^y + 3{\cdot}1^y + 1)\cdot(1{\cdot}4^y + 3{\cdot}3^y + 2)$ using standard high school algebra:

$$\begin{aligned} (2{\cdot}2^y + 3{\cdot}1^y)\cdot(1{\cdot}4^y + 3{\cdot}3^y) &= 2{\cdot}1{\cdot}2^y{\cdot}4^y + 2{\cdot}3{\cdot}2^y{\cdot}3^y + 3{\cdot}1{\cdot}1^y{\cdot}4^y + 3{\cdot}3{\cdot}1^y{\cdot}3^y \\ &= 2{\cdot}8^y + 6{\cdot}6^y + 3{\cdot}4^y + 9{\cdot}3^y. \end{aligned}$$

4. We describe the connection between the last two parts as follows. Given a polynomial p, we let $d(p)$ denote the Dirichlet series $\sum_{i\in p(1)} |p[i]|^y$. Then by (2.74),

$$\begin{aligned} d(p\otimes q) &= \sum_{i\in p(1)}\sum_{j\in q(1)} |p[i]\times q[j]|^y \\ &= \sum_{i\in p(1)} |p[i]|^y \sum_{j\in q(1)} |q[j]|^y \\ &= d(p)\cdot d(q). \end{aligned}$$

The last two parts are simply an example of this identity for a specific choice of p and q.

Solution to Exercise 2.79.

We compute $(3y^5 + 6y^2) \otimes 4$ using Exercise 2.76 #2 and the fact that $4 = 4y^0$:

$$\begin{aligned}(3y^5 + 6y^2) \otimes 4y^0 &\cong (3 \times 4)y^{5\times 0} + (6 \times 4)y^{2\times 0}\\ &\cong 12y^0 + 24y^0\\ &\cong 36.\end{aligned}$$

Solution to Exercise 2.80.

1. We show that $p \otimes y \cong p$:

$$\begin{aligned}p \otimes y &\cong \sum_{i\in p(1)} \sum_{j\in 1} y^{p[i]\times 1} && (2.74)\\ &\cong \sum_{i\in p(1)} y^{p[i]} \cong p.\end{aligned}$$

2. We show that $(p \otimes q) \otimes r \cong p \otimes (q \otimes r)$:

$$\begin{aligned}(p \otimes q) \otimes r &\cong \left(\sum_{i\in p(1)} \sum_{j\in q(1)} y^{p[i]\times q[j]}\right) \otimes r && (2.74)\\ &\cong \sum_{i\in p(1)} \sum_{j\in q(1)} \left(\sum_{k\in r(1)} y^{(p[i]\times q[j])\times r[k]}\right) && (2.74)\\ &\cong \sum_{i\in p(1)} \left(\sum_{j\in q(1)} \sum_{k\in r(1)} y^{p[i]\times(q[j]\times r[k])}\right) && (\text{Associativity of } \textstyle\sum \text{ and } \times)\\ &\cong p \otimes \left(\sum_{j\in q(1)} \sum_{k\in r(1)} y^{q[j]\times r[k]}\right) && (2.74)\\ &\cong p \otimes (q \otimes r). && (2.74)\end{aligned}$$

3. We show that $(p \otimes q) \cong (q \otimes p)$:

$$\begin{aligned}p \otimes q &\cong \sum_{i\in p(1)} \sum_{j\in q(1)} y^{p[i]\times q[j]} && (2.74)\\ &\cong \sum_{j\in q(1)} \sum_{i\in p(1)} y^{q[j]\times p[i]} && (\text{Commutativity of } \textstyle\sum \text{ and } \times)\\ &\cong q \otimes p. && (2.74)\end{aligned}$$

Solution to Exercise 2.82.

1. By Exercise 2.76 #1, we have $A \otimes B \cong AB$.
2. We use Exercise 2.76 #2 to compute $A \otimes q$:

$$A \otimes q \cong \sum_{j\in q(1)} Ay^{0\times q[i]} \cong \sum_{j\in q(1)} A \cong A \times q(1).$$

3. By Exercise 2.76 #1, we have $Ay \otimes By \cong ABy$.
4. We show that $(p \otimes q)(1)$ and $p(1) \times q(1)$ are isomorphic. By (2.74),

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

Solution to Exercise 2.83.

For each of the following classes of polynomials, we determine whether they are closed under $\otimes$ and whether they contain y.

1. The set $\{Ay^0 \mid A \in \mathbf{Set}\}$ of constant polynomials is closed under $\otimes$ by the solution to Exercise 2.82 #1. But the set does not contain y, as y is not a constant polynomial.
2. The set $\{Ay \mid A \in \mathbf{Set}\}$ of linear polynomials is closed under $\otimes$ by the solution to Exercise 2.82 #3 and does contain y, as $y \cong 1y$.
3. The set $\{Ay + B \mid A, B \in \mathbf{Set}\}$ of affine polynomials is closed under $\otimes$, for Exercise 2.76 #2 yields
$$(Ay + B) \otimes (A'y + B') \cong AA'y + AB' + BA' + BB'.$$
The set contains y, as $y \cong 1y + 0$.
4. The set $\{Ay^2 + By + C \mid A, B, C \in \mathbf{Set}\}$ of quadradic polynomials is not closed under $\otimes$, for even though $y^2 \cong 1y^2 + 0y + 0$ is a quadratic polynomial, Exercise 2.76 #1 implies that
$$y^2 \otimes y^2 \cong y^4,$$
which is not quadratic. The set contains y, as $y \cong 0y^2 + 1y + 0$.
5. The set $\{Ay^B \mid A, B \in \mathbf{Set}\}$ of monomials is closed under $\otimes$ by Exercise 2.76 #1 and does contain y, as $y \cong 1y^1$.
6. The set $\{Sy^S \mid S \in \mathbf{Set}\}$ of systematic polynomials is closed under $\otimes$, for Exercise 2.76 #1 yields
$$Sy^S \otimes Ty^T \cong STy^{ST}.$$
The set contains y, as $y \cong 1y^1$.
7. The set $\{p \in \mathbf{Poly} \mid p(1) \text{ is finite}\}$ is closed under $\otimes$ by the solution to Exercise 2.82 #4. The set contains y, as $y(1) \cong 1$ is finite.

Solution to Exercise 2.84.

The smallest class of polynomials that's closed under $\otimes$ and contains y is just $\{y\}$. This is because by Exercise 2.76 #1, we have $y \otimes y \cong y$.

Solution to Exercise 2.85.

We show that $(p_1 + p_2) \otimes q \cong (p_1 \otimes q) + (p_2 \otimes q)$ using (2.74):

$$\begin{aligned}
(p_1 + p_2) \otimes q &\cong \sum_{k \in 2} \sum_{i \in p_k(1)} \sum_{j \in q(1)} y^{p_k[i] \times q[j]} \\
&\cong \sum_{i \in p_1(1)} \sum_{j \in q(1)} y^{p_1[i] \times q[j]} + \sum_{i \in p_2(1)} \sum_{j \in q(1)} y^{p_2[i] \times q[j]} \\
&\cong (p_1 \otimes q) + (p_2 \otimes q).
\end{aligned}$$

Solution to Exercise 2.89.

1. To show that the operation $(A, B) \mapsto A + AB + B$ on **Set** is associative, we observe that
$$\begin{aligned}
(A + AB + B) + (A + AB + B)C + C &\cong A + AB + B + AC + ABC + BC + C \\
&\cong A + AB + ABC + AC + B + BC + C \\
&\cong A + A(B + BC + C) + (B + BC + C).
\end{aligned}$$
2. To show that 0 is unital for this operation, we observe that
$$(A, 0) \mapsto A + A0 + 0 \cong A$$
and
$$(0, B) \mapsto 0 + 0B + B \cong B.$$

3. We let $(1, \odot)$ denote the corresponding monoidal product on **Poly** and evaluate $(y^3+y)\odot(2y^2+2)$. By (2.88), with $A \star B \cong A + AB + B$, we have

$$\begin{aligned}(y^3 + y) \odot (2y^2 + 2) &\cong (y^3 + y^1) \odot (y^2 + y^2 + y^0 + y^0) \\ &\cong y^{3\star 2} + y^{3\star 2} + y^{3\star 0} + y^{3\star 0} + y^{1\star 2} + y^{1\star 2} + y^{1\star 0} + y^{1\star 0} \\ &\cong 2y^{11} + 2y^3 + 2y^5 + 2y^1.\end{aligned}$$

Solution to Exercise 2.95.

Here is one interpretation of the swarm-like semantics of the $\otimes$-monoid in Example 2.94. In a crew of n subordinates, each subordinate $i \in \mathsf{n}$ must determine how much of Product T to give back to the i^{th} stakeholder, who has has contributed some quantity $a_i \in \mathbb{R}_{\geq 0}$ of Resource A. The subordinates pool together the total amount $\sum_{i\in\mathsf{n}} a_i$ of Resource A that they have received, which they pass on to their supervisor. The supervisor, who sees only this total, uses it to decide (through unspecified means, perhaps via another morphism we compose with) the total quantity $t \in \mathbb{R}_{\geq 0}$ of Product T to be split between the stakeholders. Upon receiving this quantity, each subordinate then gives their stakeholder their cut $a_i t / \sum_{i\in\mathsf{n}} a_i$ of Product T proportional to their contribution of Resource A.

Solution to Exercise 2.97.

**

Solution to Exercise 2.98.

**

Solution to Exercise 2.100.

Given that m, n carry $\otimes$-monoid structures, we give an $\otimes$-monoidal structure carried by $m \otimes n$. In fact, the following works in any symmetric monoidal category. The unit $y \to m \otimes n$ is given by the tensor of the units,

$$y \cong y \otimes y \xrightarrow{\eta_m \otimes \eta_n} m \otimes n,$$

and the monoidal product is given by swapping (that's where the symmetry is used) and using the monoidal products

$$(m \otimes n) \otimes (m \otimes n) \to (m \otimes m) \otimes (n \otimes n) \xrightarrow{\mu_m \otimes \mu_n} m \otimes n.$$

The laws are easy to verify.

Solution to Exercise 2.101.

We give one of many examples of a $\otimes$-monoid structure on $\mathbb{N}y^5$. We have that $(\mathbb{N}, 0, +)$ is a monoid in $(\mathbf{Set}, 1, \times)$, so by Example 2.92, there is a $\otimes$-monoid whose carrier is $\mathbb{N}y$, whose unit is the position 0, and whose multiplication μ is given by $+$. Meanwhile, by Example 2.93, y^5 carries a canonical $\otimes$-monoid structure with a uniquely determined unit and a multiplication μ' that is the identity on positions and the diagonal on directions. So we can apply Proposition 2.99 to obtain an $\otimes$-monoid whose carrier is $\mathbb{N}y^5$, whose unit is 0, and whose multiplication is the map

$$\mathbb{N}y^5 \otimes \mathbb{N}y^5 \cong (\mathbb{N}y \otimes y^5) \otimes (\mathbb{N}y \otimes y^5) \to (\mathbb{N}y \otimes \mathbb{N}y) \otimes (y^5 \otimes y^5) \xrightarrow{\mu \otimes \mu'} \mathbb{N}y \otimes y^5 \cong \mathbb{N}y^5$$

given by $+$ on positions and the diagonal on directions. The subordinates report the sum of the natural number labels of their individual decisions to their supervisor, who selects one of 5 options for all subordinates to select.

There are other solutions; for one thing, $(\mathbb{N}, 0, +)$ is not the only monoidal structure on $\mathbb{N}$ in **Set**.

Solution to Exercise 2.103.

We give an interpretation of the swarm-like semantics of the free $\otimes$-monoid on a polynomial $p \in \mathbf{Poly}$. From Proposition 2.102, we know that the carrier of the free $\otimes$-monoid on p has as its positions lists of positions of p. So each subordinate faces a list of decisions of p, and these lists are all concatenated into a single list of decisions of p that is sent to their supervisor. For each decision of p in the list, the supervisor selects an option, which is then sent back to the subordinate who asked about that decision. Put another way, if a larger box contains a bunch of smaller boxes as subordinates, the "shorter" lists of positions from the smaller boxes are concatenated to form the "longer" list of positions for the larger box. Given a longer list of directions there, it is then chopped up into shorter lists of the appropriate lengths so that they can be dispersed back to the subordinates.

Chapter 3

Dynamical systems as polynomial morphisms

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

3.1 Moore machines

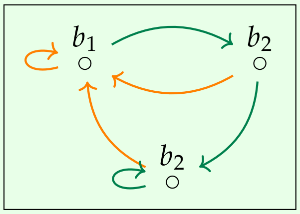
We begin with our simplest example of a dynamical system: a deterministic machine with internal states that can yield output and be updated according to input.

Definition 3.1. If A, B, and S are sets, an (A, B)-Moore machine with states S consists of two functions

$$\begin{aligned} \text{yield} &\colon S \to B \\ \text{update} &\colon S \times A \to S \end{aligned}$$

We can visualize a Moore machine as a set S of possible states. At any point in time, the machine is in one of those states: say $s \in S$. While there, whenever we ask the machine to produce output, it will give us $\text{yield}(s) \in B$. But if we feed the machine input $a \in A$, the machine will switch to a new state, $\text{update}(s, a)$.

Example 3.2. Here's a picture of a Moore machine with $S := 3$ states:



Each state is labeled by the value it yields, an 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 to make it run through its states.

Does Definition 3.1 look familiar? It's easy to see that an (A, B)-Moore machine with states S is just a lens (i.e. map of monomials)

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

with get := yield and set := update. Given such a Moore machine, we will call the monomial By^A the *interface*, because it encodes the possible inputs and outputs; and we will call the monomial Sy^S the *state system*.

Example 3.3. 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 with states $\mathbb{N}$ and interface $\mathbb{N}y$. The associated polynomial map $\mathbb{N}y^{\mathbb{N}} \to \mathbb{N}y$ is given by the identity $\mathbb{N} \to \mathbb{N}$ on positions and the function $\mathbb{N} \cong \mathbb{N} \times 1 \to \mathbb{N}$ sending $n \mapsto n + 1$ on directions.

We can generalize both Sy^S and the interface By^A in Definition 3.1, though the latter is much simpler—we'll spend much of the section on it. After some examples, we'll briefly preview something we have already hinted at and will spend a lot of time on 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 state system.

Example 3.4. 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 location in the 2-dimensional plane, and its input type is $[0, 1] \times [0, 2\pi)$, which we might think of as a command to move a certain distance in a certain direction. The map itself is given by the lens

$$\begin{array}{cc} \mathbb{R}^2 \xrightarrow{\text{yield}} \mathbb{R}^2 & \qquad \mathbb{R}^2 \times [0,1] \times [0, 2\pi) \xrightarrow{\text{update}} \mathbb{R}^2 \\ (x, y) \mapsto (x, y) & \qquad (x, y, r, \theta) \mapsto (x + r\cos\theta, y + r\sin\theta) \end{array}$$

Exercise 3.5 (Solution here). Explain in words what the Moore machine in Example 3.4 does. ◊

Example 3.6 (From functions to memoryless Moore machines). For any function $f\colon A \to B$, 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.

It is given by the map $(\mathrm{id}_B, \texttt{const}\ f)\colon By^B \to By^A$. That is, it is the identity on positions, yielding the state directly as output, and on directions it is the function $B \times A \xrightarrow{\pi} A \xrightarrow{f} B$, which ignores the current state and applies f to the input to compute the new state.

If the machine begins in state b_0 and is given a stream $(a_1, a_2, \ldots)$, the machine's outputs will be $(b_0, f(a_1), f(a_2), \ldots)$. We could say this machine is *memoryless*, because at no point does the state of the machine depend on any previous states.

Exercise 3.7 (Solution here). Suppose we are given a function $f\colon A \times B \to B$.

1. Find a corresponding (A, B)-Moore machine $By^B \to By^A$.
2. Would you say the machine is memoryless? ◊

Exercise 3.8 (Solution here). Find $A, B \in \mathbf{Set}$ such that the following can be identified with a morphism $Sy^S \to By^A$, and explain in words what the corresponding Moore machine does (there may be multiple possible solutions):

1. a *discrete dynamical system*, i.e. a set of states S and a transition function $S \to S$ that tells us how to move from state to state.
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 3.9 (Solution here). Consider the Moore machine in Example 3.4, 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 real value between 0 and 10. Make the robot lose half its health whenever it moves to a location whose x-coordinate is negative. Do not output its health; instead, use its health h as a multiplier, allowing it to move a distance of hr given an input of r. ◊

Exercise 3.10 (Solution here). Let's say a file of length n is a function $f\colon \mathsf{n} \to \mathsf{ascii}$, where $\mathsf{ascii} := \mathsf{256}$. We refer to elements of $\mathsf{n} = \{1, \ldots, n\}$ as positions in the file and, for each position $i \in \mathsf{n}$, the value $f(i)$ as the character at position i.

Given a file f, make a robot (Moore machine) whose output type is $\mathsf{ascii} + \{\text{done}\}$ and whose input type is

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

For any input, if it is of the form (s, t), then the robot should go to position s in the file and read the character at that position. If the input is "continue," the robot should move to the next position (i.e. from s to $s + 1$) and read that character—unless the new position would be greater than t, in which case the robot should continually output "done" until it receives another (s, t) pair. ◊

While Exercise 3.10 gives us a functioning file-reading robot, it is a little strange that we are still able to give the input "continue" even when the output is "done," or input a new range of positions before the robot has finished reading from the previous range. When we introduce generalized Moore machines, we will be able to let the robot "close its port," so that it can't receive signals while it's busy reading, but open its port again once it's "done"; see Example 3.42.

Exercise 3.11 (Tape of a Turing machine; solution here). 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 the "current" position. Thus the set of states of the tape is $V^{\mathbb{Z}} \times \mathbb{Z}$.

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 (i.e. decrease the current position by 1) or right (i.e. increase by 1). Thus the set of outputs of the tape is V and the set of inputs 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 $p \in$ **Poly**.
2. Write down the specific t that makes it act like a tape as specified above. ◊

3.1.1 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?

Definition 3.12. 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 3.13. A deterministic finite state automaton with a set of states S and a set of input symbols A 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 $F \subseteq S$, together with a function $u\colon S \times A \to S$, forming the rest of the required structure. □

One could also make a version of this story where, whenever the machine hits an accept-state, it stops; again, to do this requires polynomials (in this case, $y^A + 1$), which we'll get to in Section 3.2.

Niu: More about regular languages, either in this section or the next. Maybe this section should just be called "deterministic finite state automata."

3.1.2 Products: interfaces operating on the same system

One thing we can use right away in our thinking about dynamical systems is products of polynomials. Products give us a way to combine multiple polynomials into one, and in particular, the product of monomials is still a monomial. So it should not come as a surprise that they allow us to combine multiple Moore machines. What is interesting is that the behavior of this new Moore machine has a natural interpretation in terms of the behaviors of the original machines.

For any polynomials s, p_1, p_2 and maps $s \to p_1$ and $s \to p_2$, the universal property of products gives us a map $s \to p_1 p_2$. In the context of Moore machines, we have the following.

Proposition 3.14. Suppose we have 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 $(A_1 + A_2, B_1 B_2)$-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. □

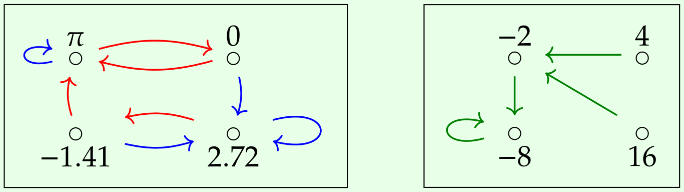
Corollary 3.15. Given $n \in \mathbb{N}$, suppose we have an (A_i, B_i)-Moore machine with state set S for each $i \in \mathsf{n}$. Then there is an induced $(\sum_{i \in \mathsf{n}} A_i, \prod_{i \in \mathsf{n}} B_i)$-Moore machine, again with state set S.

Proof. The result follows from Proposition 3.14 by induction on n. □

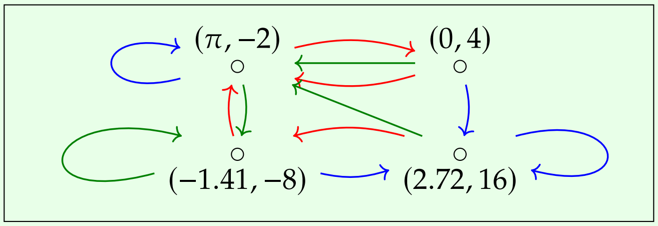
By Exercise 2.72, if each (A_i, B_i)-Moore machine has yield_i as its yield function and update_i as its update function, then the induced $(\sum_{i \in \mathsf{n}} A_i, \prod_{i \in \mathsf{n}} B_i)$-Moore machine has a yield function that sends $s \in S$ to the n-tuple $(\text{yield}_i(s))_{i \in \mathsf{n}}$ and an update function that sends $(s, a) \in S \times \sum_{i \in \mathsf{n}} A_i$ to $\text{update}_i(s, a)$ if $a \in A_i$.

In other words, if there are multiple interfaces that can drive the same set of states, we can view them as a single interface that drives the states together. This single Moore machine can receive inputs from any one of the original machines' input sets and update its state accordingly; it then yields output in all of the original machines at once. It's as though each of the machines can see where the combined system is at any time, but only one of them can actually operate it at any given time.

Example 3.16. Consider two four-state dynamical systems $e\colon 4y^4 \to \mathbb{R}y^{\{r,b\}}$ and $f\colon 4y^4 \to \mathbb{R}y^{\{g\}}$, 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:



Each state now gives two outputs: one according to the output function of e, and another according to the output function of f. As for the possible inputs, we now have the option of giving either an input from the input set of e (either r or b), in which case the machine will update its state according to the update function of e, or an input from the input set of f (only g), in which case the machine will update its state according to the update function of f.

Exercise 3.17 (Toward event-based systems; solution here). Let $f\colon Sy^S \to By^A$ 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 $f'\colon Sy^S \to p$ that has this behavior; what is p and what is f'? ◊

3.1.3 Parallel products: juxtaposing machines

Another way to combine two polynomials is by taking their parallel product, as in **??**. As with the categorical product, the parallel product of monomials is still a monomial. So parallel products give us another way to create new Moore machines from old ones by combining them.

Proposition 3.18. Suppose we have an (A_1, B_1)-Moore machine with state set S_1 and an (A_2, B_2)-Moore machine with state set S_2. Then there is an induced $(A_1 A_2, B_1 B_2)$-Moore machine with state set $S_1 S_2$.

Proof. We are given maps of polynomials $S_1 y^{S_1} \to B_1 y^{A_1}$ and $S_2 y^{S_2} \to B_2 y^{A_2}$, so we can take their parallel product to get a map

$$S_1 S_2 y^{S_1 S_2} \cong (S_1 y^{S_1}) \otimes (S_2 y^{S_2}) \to (B_1 y^{A_1}) \otimes (B_2 y^{A_2}) \cong (B_1 B_2) y^{A_1 A_2},$$

as desired. □

Corollary 3.19. Given $n \in \mathbb{N}$, suppose we have an (A_i, B_i)-Moore machine with state set S_i for each $i \in \mathsf{n}$. Then there is an induced $(\prod_{i \in \mathsf{n}} A_i, \prod_{i \in \mathsf{n}} B_i)$-Moore machine with state set $\prod_{i \in \mathsf{n}} S_i$.

Proof. The result follows from Proposition 3.18 by induction on n. □

By Proposition 2.81, if each (A_i, B_i)-Moore machine has yield_i as its yield function and update_i as its update function, then the induced $(\prod_{i \in \mathsf{n}} A_i, \prod_{i \in \mathsf{n}} B_i)$-Moore machine has a yield function that sends the n-tuple of states $(s_i)_{i \in \mathsf{n}} \in \prod_{i \in \mathsf{n}} S$ to the n-tuple of outputs $(\text{yield}_i(s))_{i \in \mathsf{n}}$ and an update function that sends $((s_i)_{i \in \mathsf{n}}, (a_i)_{i \in \mathsf{n}}) \in S \times \prod_{i \in \mathsf{n}} A_i$ to $(\text{update}_i(s_i, a_i))_{i \in \mathsf{n}}$. In other words, if there are multiple interfaces that drive the same set of states, we can view them as a single interface that drives the states together. This single Moore machine can receive inputs from any one of the original machines' input sets and update its state accordingly; it then yields output in all of the original machines at once.

3.1.4 Composing morphisms: wrapper interfaces

Example 3.20 (Paddling). Consider a Moore machine with interface $\mathbb{N}y$; its output could be any stream of natural numbers. We may interpret each natural number as the machine's current location. What if we don't want this machine to jump around wildly? Instead, suppose we want to be very strict about what how far the machine can move and what makes it move.

To accomplish this, we introduce two intermediary interfaces, which we call the *paddler* and the *tracker*:[a]

$$\textit{paddler} \coloneqq 2y \qquad \text{and} \qquad \textit{tracker} \coloneqq \mathbb{N}y^2$$

The paddler has interface $2y$ because it is blind (i.e. takes no inputs) and can only move (i.e. output) its paddle left or right: $2 \cong \{\text{left}, \text{right}\}$. The tracker has interface $\mathbb{N}y^2$ because it will announce the location of the machine (as an element $n \in \mathbb{N}$) and watch the paddler's direction (as an element of 2). Their enclosure together

$$\textit{tracker} \otimes \textit{paddler} \to \mathbb{N}y$$

is the obvious rearrangement of the identity function $\mathbb{N} \times 2 \to \mathbb{N} \times 2$, so that we can see the tracker's announcement and the tracker can see the paddler's direction.

Let's leave the paddler's dynamics alone—how you make that paddler behave is totally up to you—and instead focus on the dynamics of the tracker. We want it to watch for when the paddle switches from left to right or from right to left; at that moment it should push the paddler forward one unit. Thus the states of the tracker are given by $S \coloneqq 2\mathbb{N}$, and its dynamics

$$\varphi \colon Sy^S \to \mathbb{N}y^2$$

are given by the map given on elements $p, p' \in 2$ and $i \in \mathbb{N}$ by the formula:

$$(p, i) \mapsto i, p' \mapsto \begin{cases} i & \text{if } p = p' \\ i + 1 & \text{if } p \neq p'. \end{cases}$$

As advertised, when the paddle switches, the tracker announces that the machine has moved forward one unit; when the paddle stays still, the tracker announces that the machine itself also stays still.

---

[a] Perhaps one could refer to the tracker as the *demiurge*; it is responsible for maintaining the material universe.

Exercise 3.21 (Solution here). Change the dynamics and state-set of the tracker in Example 3.20 so that it exhibits the following behavior.

When the paddle switches once and stops, the tracker increases its position by one unit and stops, as before in Example 3.20. But when the paddle switches twice in a row, the tracker increases its position by two units on the second switch! So if it is quiet for a while and then switches three times in a row, the tracker will increase its position by one then two then two. ◊

3.1.5 Situations

Let p be a polynomial. We will refer to a map $p \to y$ as a *situation* for p, and denote the set of them by

$$\Gamma(p) := \mathbf{Poly}(p, y) \tag{3.22}$$

as we did in Proposition 1.21. By (2.42), we have that

$$\Gamma(p) \cong \prod_{i \in p(1)} p[i], \tag{3.23}$$

so a situation $\gamma \in \Gamma(p)$ can be thought of as a dependent function that takes every position $i \in p(1)$ and returns a direction $\gamma(i) \in p[i]$. The idea for the name is that a situation dictates what you'll see (the input you receive), given anything you might do (the output you provide). If the situation is that you're at a party, then that dictates what direction you'll receive, given what position you take.

A map $p_1 \otimes \cdots \otimes p_n \to y$ puts these n interfaces in a situation together. The following proposition helps explain why situations are a useful concept.

Proposition 3.24. Given polynomials $p, q \in \mathbf{Poly}$, there is a bijection

$$\Gamma(p \otimes q) \cong \mathbf{Set}\big(q(1), \Gamma(p)\big) \; \times \; \mathbf{Set}\big(p(1), \Gamma(q)\big). \tag{3.25}$$

The idea is that for p and q to be enclosed together in a system simply means that the positions of each one become a situation for the other.

Proof of Proposition 3.24. This is a direct calculation:

$$\begin{aligned}
\Gamma(p \otimes q) &\cong \prod_{i \in p(1)} \prod_{j \in q(1)} (p[i] \times q[j]) \\
&\cong \left(\prod_{j \in q(1)} \prod_{i \in p(1)} p[i] \right) \times \left(\prod_{i \in p(1)} \prod_{j \in q(1)} q[j] \right) \\
&\cong \mathbf{Set}(q(1), \Gamma(p)) \times \mathbf{Set}(p(1), \Gamma(q)).
\end{aligned}$$

□

Exercise 3.26 (Solution here).

1. State and prove a generalization of (3.25) from Proposition 3.24 for n-many polynomials $p_1, \ldots, p_n \in \mathbf{Poly}$.
2. Generalize the "idea" statement between Proposition 3.24 and its proof. ◊

Exercise 3.27 (Solution here). We will use (3.25) to consider the interaction φ between *you* and *chalk* from Example 3.44 as a pair of functions $\mathit{you}(1) \to \Gamma(\mathit{chalk})$ and $\mathit{chalk}(1) \to \Gamma(\mathit{you})$.

1. How is the chalk's position a situation for you? That is, write the map $\mathit{chalk}(1) \to \Gamma(\mathit{you})$.
2. How is your position a situation for the chalk? That is, write the map $\mathit{you}(1) \to \Gamma(\mathit{chalk})$. ◊

Proposition 3.28. The situations functor $\Gamma\colon \mathbf{Poly} \to \mathbf{Set}^{\mathrm{op}}$ sends $(0, +)$ to $(1, \times)$:

$$\Gamma(0) \cong 1 \qquad \text{and} \qquad \Gamma(p + q) \cong \Gamma(p) \times \Gamma(q).$$

Technically, one could say that Γ preserves coproducts, since coproducts in $\mathbf{Set}^{\mathrm{op}}$ are products in $\mathbf{Set}$.

Exercise 3.29 (Solution here). Prove Proposition 3.28. ◊

The situations functor $\Gamma\colon \mathbf{Poly} \to \mathbf{Set}^{\mathrm{op}}$ is also normal lax monoidal in the sense that there are canonical functions

$$1 \cong \Gamma(y) \qquad \text{and} \qquad \Gamma(p) \times \Gamma(q) \to \Gamma(p \otimes q)$$

satisfying certain well-known laws. We won't prove this unless or until we need it.

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

$$Sy^S \to p$$

for some $S \in \mathbf{Set}$ and $p \in \mathbf{Poly}$. 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 $p \coloneqq By^A$ for sets A, B. For the standard Moore machines that we have been working with, the set of inputs was always fixed; if the interface were By^A, the set of inputs would always be A.

On the other hand, an arbitrary polynomial looks like $p \coloneqq \sum_{j \in J} B_j y^{A_j}$. So the output of a dependent system with interface p is an element of the coproduct $\sum_{j \in J} B_j$. Then which B_j the output is in determines the sort of input A_j you get. The set of inputs is no longer constant; it varies depending on the current output. What kind of system has that kind of a relationship between its inputs and outputs?

Well, we can begin to think of outputs not only as outward expressions, but as positions that one takes within one's arena—terminology we've been using all along. If the arena is your body, then your outputs would be the positions that your body could take. 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, every form of output you provide, from talking to gesturing to moving another object, 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 itself a sensory organ: a hand outstretched, an eye open or closed.

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 mathematically. Let's start with something familiar.

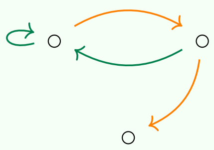
Example 3.31. Recall regular languages from Definition 3.12. Often one does not want to "keep going" after recognizing a word in one's language. For that, rather than use a map $Sy^S \to \mathsf{2}y^A$, we could use a map

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

To give such a map, one in particular provides a function $f_1\colon S \to \mathsf{2}$; here we are thinking of $\mathsf{2}$ as the set of positions of $y^A + \mathsf{1}$. 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 $\mathsf{1}$, no inputs available there. In other words, the function $f^\sharp$ is trivial there.

On the other hand, $f^\sharp_s$ is not trivial on those elements $s \in S$ for which $f_1(s) = 1$. They must come equipped with a function $f^\sharp_s\colon A \to S$, 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 3.32 (Solution here). 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 3.31.

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 3.33 (Graphs). Given a graph $G = (E \rightrightarrows V)$ with source map $s\colon E \to V$ and target map $t\colon E \to V$, 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 $f\colon Vy^V \to g$, where $f_1 = \mathrm{id}_V$ and $f^\sharp(v, e) = t(e)$.

Exercise 3.34 (Solution here). Pick your favorite graph G, and consider the associated dynamical system as in Example 3.33. Draw the associated labeled transition system as in Example 3.16. ◊

Example 3.35 (Adding a pause button). Given any dependent dynamical system $f\colon Sy^S \to p$, 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 $\epsilon\colon Sy^S \to y$ given by identity $\mathrm{id}_S\colon S \to S$ (see Exercise 3.36). By the universal property of products, we can pair f and ϵ to get a map $(f, \epsilon)\colon Sy^S \to py \cong p \times y$. This process is actually universal in a way we'll find important later. Indeed, it's called *copointing*; see Proposition 7.20.

Exercise 3.36 (Solution here). What does it mean in Example 3.35 that the map $\epsilon\colon Sy^S \to y$ is given by the identity id_S? ◊

Example 3.37 (Repeater). Suppose given a dependent dynamical system φ that sometimes outputs an element of A and sometimes outputs only silence. That is, its interface is $Ay + y$. But what if some other system inputs only A's; we will combine φ with another system—a repeater—to get an A-emitter, i.e. a system with interface Ay. How do we construct the repeater, and how do the two systems fit together?

For the repeater, we will dispense with any interesting interface and just use Ay^A: it listens for an A and outputs an A. We enclose the two systems in an A-emitter using a map

$$\psi\colon (Ay + y) \otimes Ay^A \to Ay$$

which we now describe. Since $\otimes$ distributes over $+$, it suffices to give maps $Ay \otimes Ay^A \to Ay$ and $y \otimes Ay^A \to Ay$. The former corresponds to the case that the system is currently outputting, and the second corresponds to when the system is silent. For the former we use the map

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

In other words, we output the repeater's position as the current A, and we update the repeater's position to the system's current state.

For the latter it suffices—by the universal property of products—to give $Ay^A \to y$ and $Ay^A \to A$. For the first we use ϵ, which means that during silence, the repeater holds the A constant; for the second we use the product projection, which means that we output the current value of the repeater.

Example 3.38 (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.

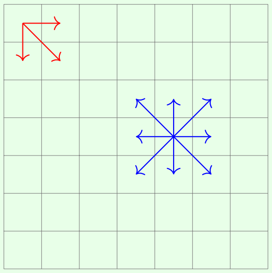
To give a morphism $S'y^{S'} \to y + y^S$, it is sufficient to give two morphisms: $Sy^{S'} \to y$ and $y^{S'} \to y^S$. The first is equivalent to a function $S \to S'$ and we take the map $S \xrightarrow{f^\sharp} S \xrightarrow{i} S'$; this means that whenever we want to update the state from a state 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 3.39 (Solution here). Find what you think is an interesting generalization of deterministic finite state automata that you can model using generalized Moore machines. ◊

Example 3.40. Choose $n \in \mathbb{N}$, a *grid size*, and for each $i \in \mathsf{n} = \{1, \ldots, n\}$ 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 3.41 (Solution here). Add to Example 3.40 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 3.42. In Exercise 3.10 one is tasked with making a file reader FileReader, where a file is a function $f \colon \mathsf{n} \to \mathsf{ascii}$. Now we take that same idea but give the robot a different interface when it is in read-mode: namely, one where it cannot take in signals.

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

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

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

$$\begin{aligned} \mathsf{In}_{\mathsf{FileReader}}(\text{Accepting}(c)) &= \mathsf{State}_{\mathsf{FileReader}}, \\ \mathsf{In}_{\mathsf{FileReader}}(\text{Busy}(c)) &= \mathsf{1}. \end{aligned}$$

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

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

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.

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

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

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

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

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

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

Exercise 3.43 (Solution here). Make a file reader that acts like that in Example 3.42, except that it only emits output $o \in \mathsf{ascii}$ when $o = 100$. ◊

Example 3.44 (Picking up the chalk). Imagine you are at a table; you see some chalk and you pinch it between your thumb and forefinger. An amazing thing about reality is that you will then have the chalk, in the sense that you can move it around. How might we model this in **Poly**?

Let's say that your hand can be at one of two heights, down or up, and that you can either press (apply pressure between your thumb and forefinger) or not press. Let's also say that you take in information about the chalk's height. Here are the two sets we'll be using:

$$H \coloneqq \{\text{down}, \text{up}\} \qquad \text{and} \qquad P \coloneqq \{\text{press}, \text{no press}\}.$$

Your interface is HPy^H: producing your own height and pressure, and receiving the chalk's height.

As for the chalk, it is in one of two modes: in your possession or not. Either way, it reveals its height, which is either down on the table or up in the air. The chalk always takes in information about whether pressure is being applied or not. When it's out of your possession (mode "out"), that's the whole story, but when it is in your possession (mode "in") it also receives your hand's height. All together, here are the two interfaces:

$$\mathit{you} \coloneqq HPy^H \qquad \text{and} \qquad \mathit{chalk} \coloneqq \{\text{out}\}Hy^P + \{\text{in}\}Hy^{HP}.$$

Now we want to give the interaction between you and the chalk. As we said before, you see the chalk's height. If your hand is not at the height of the chalk, the chalk receives no pressure. Otherwise, your hand is at the height of the chalk, so the chalk receives your pressure (or lack thereof). Furthermore, if the chalk is in your possession, it also receives your hand's height.

To provide a map $\mathit{you} \otimes \mathit{chalk} \xrightarrow{\varphi} y$, we use the fact that *chalk* is a sum and that $\otimes$ distributes over $+$. Thus we need to give two maps

$$HPy^H \otimes Hy^P \xrightarrow{\psi} y \qquad \text{and} \qquad HPy^H \otimes Hy^{HP} \xrightarrow{\psi'} y$$

The map ψ', corresponding to when the chalk is in your possession, is quite easy to describe; it can be unfolded to a function $HPH \to HHP$, and we take it to be the obvious map sending your height and pressure to the chalk and the chalk's height to you; see Exercise 3.46. But ψ is more semantically interesting: it is given by the map

$$((h_{\mathit{you}}, p_{\mathit{you}}), h_{\mathit{chalk}}) \mapsto \begin{cases} (h_{\mathit{chalk}}, \text{no press}) & \text{if } h_{\mathit{you}} \neq h_{\mathit{chalk}} \\ (h_{\mathit{chalk}}, p_{\mathit{you}}) & \text{if } h_{\mathit{you}} = h_{\mathit{chalk}}. \end{cases}$$

So now we've got you and the chalk in a closed system together, given by $\varphi = \psi + \psi'$, so we are ready to add some dynamics. Your dynamics can be whatever you want, so let's just add some dynamics to the chalk and call it a day. The chalk has only four states $C \coloneqq \{\text{out}, \text{in}\} \times H \cong 4$: the H coordinate is its current height, and the other coordinate is whether or not it is "in your possession." We will give a dynamical system $Cy^C \to \mathit{chalk}$ with states C and chalk-interface, i.e. a map

$$\{\text{out}, \text{in}\} \times Hy^{\{\text{out},\text{in}\}\times H} \to \{\text{out}\}Hy^P + \{\text{in}\}Hy^{HP}. \tag{3.45}$$

As you might guess, the chalk reveals its height and whether it is in your possession directly. If it's not in your possession, it falls down unless you catch it (i.e. apply pressure to it so that it enters your possession); if it is in your possession, it takes whatever height you give it. This is all expressed by the following dynamics, which define (3.45):

$$\begin{aligned}
(\text{out}, h_{\mathit{chalk}}) &\mapsto \left(\text{out}, h_{\mathit{chalk}}, \begin{array}{r} \text{no press} \mapsto (\text{out}, \text{down}) \\ \text{press} \mapsto (\text{in}, h_{\mathit{chalk}}) \end{array}\right) \\
(\text{in}, h_{\mathit{chalk}}) &\mapsto \left(\text{in}, h_{\mathit{chalk}}, \begin{array}{r} (h_{\mathit{you}}, \text{no press}) \mapsto (\text{out}, h_{\mathit{you}}) \\ (h_{\mathit{you}}, \text{press}) \mapsto (\text{in}, h_{\mathit{you}}) \end{array}\right)
\end{aligned}$$

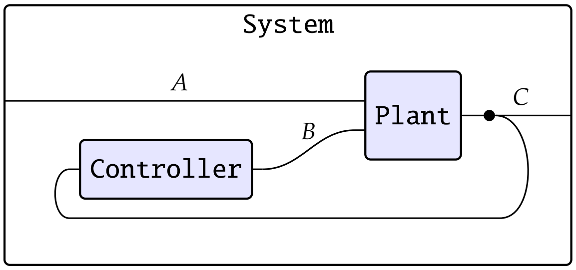
Obviously, this is all quite complicated, intricate, and contrived. Our goal here is only to show that you can define interactions in which one system can engage with or disengage from another, and that when the two systems are engaged, the first controls the behavior of the second.

Exercise 3.46 (Solution here).

1. In Example 3.44, we said that $\psi' \colon HPy^H \otimes Hy^{HP} \to y$ was easy to describe and given by a function $HPH \to HHP$. Explain what's being said, and provide the function.
2. Provide dynamics to the *you* character so that you repeatedly reach down and grab the chalk, lift it with your hand, and drop it. ◊

3.3 Wiring diagrams

We want our dynamical systems to interact with each other.



(3.47)

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 (3.47): the controller, the plant, and the system. Each has inputs and outputs, and so we can consider the interface as a monomial.

$$\texttt{Plant} = Cy^{AB} \qquad \texttt{Controller} = By^{C} \qquad \texttt{System} = Cy^{A}. \tag{3.48}$$

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

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

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 C; 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 *set* 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 3.49 (Solution here).

1. Make a new wiring diagram like (3.47) except where the controller also receives information of type A' from the outside world.
2. What are the monomials in your diagram (replacing (3.48))?
3. What is the morphism of polynomials corresponding to this diagram? ◊

Now suppose given a dynamical system in each inner box:

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

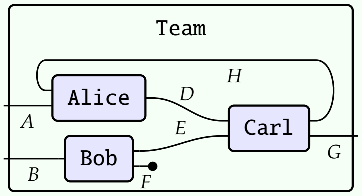
Then since ⊗ is a monoidal product on **Poly** (see Proposition 2.81), we get a map

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

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

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

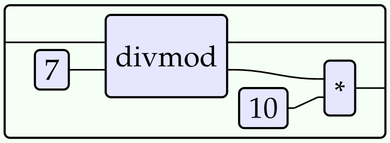
Exercise 3.50 (Solution here). Consider the following wiring diagram.



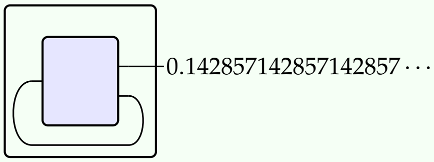
1. Write out the polynomials for each of Alice, Bob, and Carl.
2. Write out the polynomial for the outer box, Team.
3. The wiring diagram constitutes a morphism f in **Poly**; what is its type $f\colon ? \to ?$
4. What morphism is it?
5. Suppose we are given dynamical systems $Ay^A \to$ `Alice`, $By^B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

Exercise 3.51 (Long division; solution here).

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 3.7 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.



◊

Example 3.52 (Graphs as interaction diagrams). Suppose given a graph G and a set $\tau(v)$ for every vertex

$$\begin{tikzcd} A \arrow[r, shift left, "\mathrm{src}"] \arrow[r, shift right, "\mathrm{tgt}"'] & V \arrow[r, "\tau"] & \mathbf{Set} \end{tikzcd}$$

For each vertex $v \in V$, let $A_v := \mathrm{src}^{-1}(v) \subseteq A$ denote the arrows emanating from v, and define the monomial

$$p_v \coloneqq \tau(v) y^{\prod_{a \in A_v} \tau(\text{tgt } a)}$$

From this data we can give a morphism

$$\bigotimes_{v \in V} p_v \xrightarrow{\varphi} y.$$

To do so, we just need a function of the form

$$\prod_{v \in V} \tau(v) \longrightarrow \prod_{v \in V} \prod_{a \in A_v} \tau(\text{tgt } a)$$

We use $f \mapsto v \mapsto a \mapsto f(\text{tgt } a)$.

The map φ wires together the interfaces p_v, over all vertices $v \in V$, inside a closed outer box. That way, once we instantiate each p_v with a dynamical system $Sy^S \to p_v$—one that outputs $\tau(v)$ and inputs $\prod_{a \in A_v} \tau(\text{tgt } a)$—they will all interact together as specified by the graph.

For example, a common graph found in cellular automata is a 2-dimensional integer lattice, with vertices $V \coloneqq \mathbb{Z} \times \mathbb{Z}$. The *stencil* indicates which vertices "hear" which other vertices. One might use

$$A \coloneqq \{-1, 0, 1\} \times \{-1, 0, 1\} \times V$$

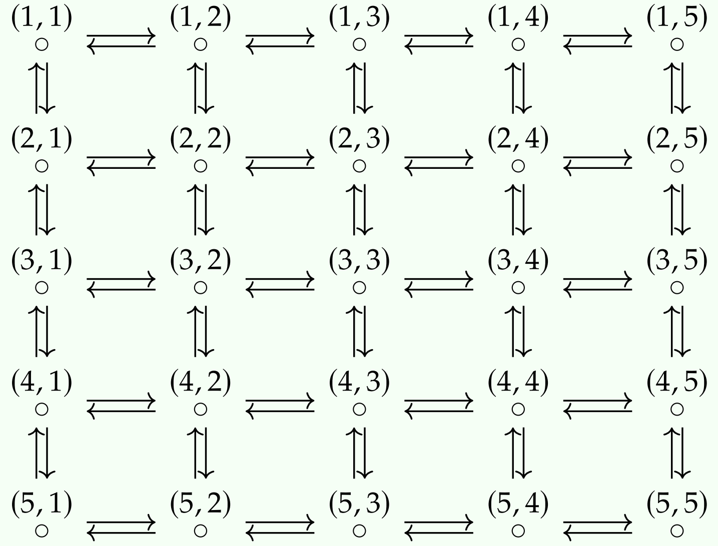
with $\text{src}(i, j, m, n) = (m, n)$ and $\text{tgt}(i, j, m, n) = (m + i, n + j)$.

Exercise 3.53 (Conway's Game of Life; solution here). This exercise is for those who are familiar with Conway's Game of Life. We will use Example 3.52.

1. What is the appropriate graph $A \rightrightarrows V$?
2. What is the appropriate function $\tau \colon V \to \mathbf{Set}$?
3. What are the polynomials p_v from Example 3.52?
4. What is the appropriate state set S_v for each interface p_v?
5. What is the appropriate dynamical system map $S_v y^{S_v} \to p_v$? ◊

Exercise 3.54 (Cellular automata; solution here). Let $G = (V, E)$ be a simple graph, i.e.

$V, E \in \mathbf{Set}$ 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

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

For each $v \in V$, let $p_v := \mathsf{2}y^{\mathsf{2}^{I(v)}}$; 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? ◊

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 3.55. 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:

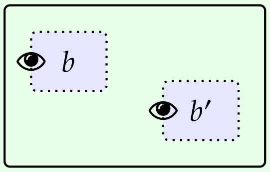
$$p \otimes p' \cong \sum_{(m,m') \in MM'} B_m B'_m y^{A_m A'_m}$$

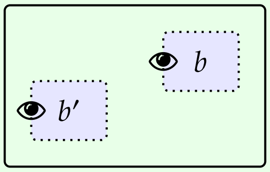
so the interface $B_m B'_{m'} y^{A_m A'_{m'}}$ at each of these $(M \times M')$-many modes (m, m') must be accommodated in any morphism $w \colon p \otimes p' \to q$. For example if $M = 2$ and $M' = 3$ then w can be specified by six maps.

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 3.56. 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').

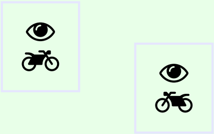
Let's think of elements of B and B' not as outputs, but as positions.





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 3.57. Suppose you have two systems p, q, both having the same 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

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

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

On directions, we use the function

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

so that 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:

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

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

Exercise 3.58 (Solution here). Let $p, q := \mathbb{N}y^{\mathbb{N}}$.

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 3.59 (Solution here). 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 $p := AXy^{A'X}$ and $p' := A'Xy^{AX}$, 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 3.60 (Cellular automata who vote on their interaction pattern). Recall from Exercise 3.54 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 3.54, 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

$$\bigotimes_{v \in V} p_v \cong \mathsf{2}^V y^{\mathsf{2}^{\sum_{v \in V} \mathsf{i}(v)}} \longrightarrow y. \tag{3.61}$$

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 $I\colon 2^V \to (2^V)^V$. 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 3.62 (Solution here). Change Example 3.60 slightly by changing the outer box.

1. First change it to Ay for some set A of your choice, and update (3.61) so that the system outputs some aspect of the current state 2^V.
2. What would it mean to change (3.61) to a map $\bigotimes_{v \in V} p_v \to y^A$ for some A? ◊

Example 3.63. Recall the picture from Example 1.10. 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 $F(FF) \to \{\mathtt{Just}\}F + \{\mathtt{Nothing}\}$, let's use the map $(f, (f_L, f_2)) \mapsto t(f_L, f_2)$ defined here:

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

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 3.64. 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 3.65. When someone assembles a machine, their own outputs dictate the connection pattern of the machine's components.

(3.66)

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 (3.66) 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 3.65. 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 it's 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 be constantly choosing interaction patterns. Let's see how it works.

3.5 Closure of ⊗

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

$$\mathbf{Poly}(p \otimes q, r) \cong \mathbf{Poly}(p, [q, r]) \tag{3.67}$$

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

$$[q, r] \coloneqq \prod_{j \in q(1)} r \circ (q[j]y) \tag{3.68}$$

Here $\circ$ denotes standard functor composition; informally, $r \circ (q[j]y)$ is the polynomial you get when you replace each appearance of y in r by $q[j]y$. Composition, together with the unit y, is in fact yet another monoidal structure, as we will see in more depth in Part II.

Before we prove that the isomorphism (3.67) holds naturally, let us investigate the properties of the closure operation, starting with some simple examples.

Exercise 3.69 (Solution here). 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. $q \coloneqq A$ for $A \in \mathbf{Set}$ (constant) and r arbitrary.
5. $q \coloneqq Ay$ for $A \in \mathbf{Set}$ (linear) and r arbitrary.
6. $q \coloneqq y^2 + 2y$ and $r \coloneqq 2y^3 + 3$. ◊

Exercise 3.70 (Solution here). Show that for any polynomials p_1, p_2, q, we have an isomorphism

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

◊

Exercise 3.71 (Solution here). Show that there is an isomorphism

$$[q, r] \cong \sum_{f\colon q \to r} y^{\sum_{j \in q(1)} r[f_1(j)]} \tag{3.72}$$

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

Exercise 3.73 (Solution here). Verify that (2.55) holds. ◊

Example 3.74. For any $A \in \mathbf{Set}$ we have

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

More generally, for any polynomial $p \in \mathbf{Poly}$ we have

$$[p, y] \cong \Gamma(p) y^{p(1)}. \tag{3.75}$$

All these facts follow directly from (3.68).

Exercise 3.76 (Solution here). Verify the three facts above. ◊

Exercise 3.77 (Solution here). 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 3.78. With $[-,-]$ as defined in (3.68), there is a natural isomorphism

$$\mathbf{Poly}(p \otimes q, r) \cong \mathbf{Poly}(p, [q, r]). \tag{3.79}$$

Proof. We have the following chain of natural isomorphisms:

$$\begin{aligned}
\mathbf{Poly}(p \otimes q, r) &\cong \mathbf{Poly}\Big(\sum_{i \in p(1)} \sum_{j \in q(1)} y^{p[i]q[j]}, r \Big) \\
&\cong \prod_{i \in p(1)} \prod_{j \in q(1)} \mathbf{Poly}(y^{p[i]q[j]}, r) && \text{(Universal property of coproducts)} \\
&\cong \prod_{i \in p(1)} \prod_{j \in q(1)} r(p[i]q[j]) && \text{(Yoneda lemma)} \\
&\cong \prod_{i \in p(1)} \prod_{j \in q(1)} \mathbf{Poly}(y^{p[i]}, r \circ (q[j]y)) && \text{(Yoneda lemma)} \\
&\cong \mathbf{Poly}\Big(\sum_{i \in p(1)} y^{p[i]}, \prod_{j \in q(1)} r \circ (q[j]y) \Big) \\
& && \text{(Universal property of (co)products)} \\
&\cong \mathbf{Poly}(p, [q, r]).
\end{aligned}$$

□

Exercise 3.80 (Solution here). 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 (3.68), or just use (3.79) with the Yoneda lemma and the fact that $y \otimes p \cong p$. ◊

The closure of $\otimes$ implies that for any $p, q \in \mathbf{Poly}$, there is a canonical *evaluation* map

$$\mathrm{eval}\colon [p, q] \otimes p \longrightarrow q. \tag{3.81}$$

Exercise 3.82 (Solution here).

1. Obtain the evaluation map $\mathrm{eval}\colon [p, q] \otimes p \longrightarrow q$ from (3.81).
2. Show that for any $p, q, r \in \mathbf{Poly}$ 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:

$$\begin{tikzcd}
p \otimes q \arrow[r, "f' \otimes q"] \arrow[rr, bend right, "f"'] & {[q, r] \otimes q} \arrow[r, "\mathrm{eval}"] & r
\end{tikzcd}$$

◊

Exercise 3.83 (Solution here).

1. For any set S, obtain the map $Sy^S \to y$ whose on-directions map is the identity on S using eval and Example 3.74.
2. Show that maps of the four types κ_{11}, κ_{12}, κ_{21}, κ_{22} shown in Example 3.63 can be obtained by tensoring together identity maps and eval maps. ◊

Example 3.84 (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 $[p, r]$.

Example 3.85 (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 we are 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 $\psi_1 \times \psi_2\colon p_1 \times p_2 \to [q_1, r] \times [q_2, r]$. Then we apply Exercise 3.70 to see that $[q_1, r] \times [q_2, r] \cong [q_1 + q_2, r]$, and finally monoidal-uncurry to obtain $(p_1 \times p_2) \otimes (q_1 + q_2) \to r$ as desired.

3.6 Exercise solutions

Solution to Exercise 3.5.

At any point in time, the Moore machine in Example 3.4 is located somewhere on the 2-dimensional plane, say at the coordinates $(x, y) \in \mathbb{R}^2$, as recorded by its state. Whenever we ask the machine to produce output, it will tell us those coordinates, since $\text{yield}(x, y) = (x, y)$. But if we give the machine input of the form (r, θ) for some distance $r \in [0, 1]$ and direction $\theta \in [0, 2\pi)$, the machine will move by that distance, in that direction, going from (x, y) to

$$\text{update}(x, y, r, \theta) = (x, y) + r(\cos\theta, \sin\theta)$$

(here we treat $\mathbb{R}^2$ as a vector space, so that $r(\cos\theta, \sin\theta)$ is a vector of length r in the direction of θ).

Solution to Exercise 3.7.

1. We seek an (A, B)-Moore machine $By^B \to By^A$ corresponding to the function $f\colon A \times B \to B$. We know that an (A, B)-Moore machine $By^B \to By^A$ consists of a function $\text{yield}\colon B \to B$ and a function $\text{update}\colon B \times A \to B$. So we can simply let the yield function be the identity on B and the update function be $B \times A \cong A \times B \xrightarrow{f} B$, i.e. the function f with its inputs swapped.

2. Generally, such a machine is not memoryless. Unlike in Example 3.6, the update function $B \times A \cong A \times B \xrightarrow{f} B$ does appear to depend on its first input, namely the previous state, which f takes as its second input.
   However, if f factors through the projection $\pi\colon A \times B \to A$, i.e. if f can be written as a composite $A \times B \xrightarrow{\pi} A \xrightarrow{f'} B$ for some $f'\colon A \to B$, then the resulting machine *is* memoryless: it is the memoryless Moore machine corresponding to f', as in Example 3.6.

Solution to Exercise 3.8.

For each of the following constructs, we find $A, B \in \mathbf{Set}$ such that the construct can be identified with a morphism $Sy^S \to By^A$, i.e. a function yield: $S \to B$ and a function update: $S \times A \to S$.

1. Given a discrete dynamical system, consisting of a set of states S and a transition funtion $n\colon S \to S$, we can set $A \coloneqq B \coloneqq \mathsf{1}$; so yield: $S \to \mathsf{1}$ is unique, while update: $S \times \mathsf{1} \to S$ is given by $S \times 1 \cong S \xrightarrow{n} S$. The corresponding Moore machine has a single choice of input (you could think of it as a button that says "advance to the next state") and always produces the same output (which effectively tells us nothing). So it is just a set of states, and a deterministic way to move from state to state.
   We could have also set $A \coloneqq \mathsf{0}$ and $B \coloneqq S$, so that yield $\coloneqq n$ and update: $S \times \mathsf{0} \to S$ is unique, but this formulation is somewhat less satisfying: this is a Moore machine that never moves between its states, effectively functioning as a lookup table between whatever state the machine happens to be in and its output, which happens to refer to some state.
2. Given a magma, consisting of a set S and a function $m\colon S \times S \to S$, we can set $A \coloneqq S$ and $B \coloneqq 1$. Then yield: $S \to \mathsf{1}$ is unique, while update: $S \times S \to S$ is equal to m. The corresponding Moore machine always produces the same output. It uses the binary operation m to combine the current state with the input—which also refers to a state—to obtain the new state.
   Note that, since m is not guaranteed to be commutative, we could also set the update function to be m with its inputs swapped. The difference here is that the new state is given by applying m with the input on the left and the current state on the right, rather than the other way around.
   We could have also set $A \coloneqq \mathsf{0}$ and $B \coloneqq S^S$, so that update: $S \times \mathsf{0} \to S$ is unique, while currying m gives yield, so that yield(s) is the function $S \to S$ given by $s' \mapsto m(s, s')$. Alternatively, yield(s) could be the function $s' \mapsto m(s', s)$. Either way, this is a Moore machine that never moves between its states, functioning as a lookup table between the machine's current state and the function m partially applied to that state on one side or the other.
3. Given a set S and a subset $S' \subseteq S$, we can set $A \coloneqq \mathsf{0}$ and $B \coloneqq \mathsf{2}$. Then update: $S \times \mathsf{0} \to S$ is unique, while we define yield: $S \to 2$ by
   $$\text{yield}(s) = \begin{cases} 1 & \text{if } s \in S' \\ 2 & \text{if } s \notin S', \end{cases}$$
   so that S' can be recovered from the yield function as its fiber over 1. Alternatively, we could define the yield function so that S' is instead its fiber over 2. The corresponing Moore machine never moves between its states, but gives one of two outputs indicating whether or not the current state is in the subset S'.

Solution to Exercise 3.9.

We modify the Moore machine from Example 3.4 as follows. The original Moore machine had a state set $\mathbb{R}^2$, so to add a health meter with values in $[0, 10]$, we take the Cartesian product to obtain the new state set $\mathbb{R}^2 \times [0, 10]$. The inputs and outputs are unchanged, so the Moore machine is a lens

$$\mathbb{R}^2 \times [0, 10] y^{\mathbb{R}^2 \times [0,10]} \to \mathbb{R}^2 y^{[0,1] \times [0, 2\pi)}.$$

Its yield function $\mathbb{R}^2 \times [0, 10] \to \mathbb{R}^2$ is the canonical projection, as the machine only outputs its location in $\mathbb{R}^2$ and not its health; while its update function

$$\mathbb{R}^2 \times [0, 10] \times [0, 1] \times [0, 2\pi) \to \mathbb{R}^2 \times [0, 10]$$

sends (x, y, h, r, θ) to

$$(x + hr\cos\theta, y + hr\sin\theta, h'),$$

where $h' = h/2$ if the machine's new x-coordinate $x + hr\cos\theta < 0$ and $h' = h$ otherwise.

Solution to Exercise 3.10.

Given a file $f\colon \mathsf{n} \to \mathsf{ascii}$, we construct our file-reading robot as a Moore machine as follows. There are many options for what states the machine should record, but we will use pairs of values $(i, t) \in \mathsf{n}^2$, where i tracks the robot's current position in the file, while t tracks the position where the robot should stop. But we will also include a "done" state to record when the robot has passed the stop position. So the Moore machine is a lens

$$(\mathsf{n}^2 + \{\text{done}\})y^{\mathsf{n}^2+\{\text{done}\}} \to (\mathsf{ascii} + \{\text{done}\})y^{\{(s,t) \mid 1 \leq s \leq t \leq n\}+\{\text{continue}\}}.$$

If the robot's current state is "done," then the robot should output "done." Otherwise, the robot should output the character at its current position in the file. So its yield function $\mathsf{n}^2 + \{\text{done}\} \to \mathsf{ascii} + \{\text{done}\}$ sends (i, t) to $f(i)$ and "done" to "done." Meanwhile, the update function

$$(\mathsf{n}^2 + \{\text{done}\}) \times (\{(s, t) \mid 1 \leq s \leq t \leq n\} + \{\text{continue}\}) \to \mathsf{n}^2 + \{\text{done}\}$$

sends any state with input (s, t) to the state (s, t). On the other hand, if the input is "continue," an old state (i, t) is sent to the new state $(i + 1, t)$ if $i + 1 \leq t$ and "done" otherwise. Finally, if the old state is "done" and the input is "continue," the new state is still "done."

Solution to Exercise 3.11.

1. A Turing machine has states $V^{\mathbb{Z}} \times \mathbb{Z}$, outputs V, and inputs $V \times \{L, R\}$, so as a Moore machine, it is a map of polynomials
$$t\colon (V^{\mathbb{Z}} \times \mathbb{Z})y^{V^{\mathbb{Z}}\times\mathbb{Z}} \to Vy^{V\times\{L,R\}}.$$
2. The yield function of t should output the value at the current position of the tape. So yield: $V^{\mathbb{Z}} \times \mathbb{Z} \to V$ is the evaluation map: it sends (f, c) with $f\colon \mathbb{Z} \to V$ and $c \in \mathbb{Z}$ to $f(c)$. Meanwhile, the update function of t should write the input value of V at the current position of the tape, then move according to whether the second input value is L (left) or R (right). So
$$\text{update}\colon (V^{\mathbb{Z}} \times \mathbb{Z}) \times (V \times \{L, R\}) \to V^{\mathbb{Z}} \times \mathbb{Z}$$
sends old tape $f\colon \mathbb{Z} \to V$, old position $c \in \mathbb{Z}$, new value $v \in V$, and direction $D \in \{L, R\}$ to the new tape $f'\colon \mathbb{Z} \to V$ satisfying
$$f'(n) = \begin{cases} v & \text{if } n = c \\ f(n) & \text{if } n \neq c \end{cases}$$
and the new position $c - 1$ if $D = L$ and $c + 1$ if $D = R$.

Solution to Exercise 3.17.

Given a Moore machine $f\colon Sy^S \to By^A$, we seek a new machine $f'\colon Sy^S \to p$ that has the added option to provide no input at a step so that the machine does not change. We can think of this as having two different interfaces acting on the same system: the original interface By^A of f, and a new interface with only one possible input—namely the option to provide no input at all—that does not change the system. This latter interface also does not need to distinguish between its outputs; it should have just one possible output that says nothing. So the interface is y.

If y were the only interface acting on the system, we would have a Moore machine $g\colon Sy^S \to y$ whose yield function is the unique function $S \to 1$ and whose update function is the identity function on S, since the input never changes the system. Then p is the product of the two interfaces By^A and y, while $f'\colon Sy^S \to p$ is the unique map induced by $f\colon Sy^S \to By^A$ and $g\colon Sy^S \to y$. In particular, $p \cong By^{A+1}$, while f' consists of a yield function $S \to B$ that is the same as the yield function of f and an update function $S \times (A + 1) \to S$ that behaves like the update function of f when the input is from A but does not change the state when the input is from 1.

Solution to Exercise 3.21.

**

Solution to Exercise 3.26.

1. We generalize (3.25) for n polynomials as follows. Given polynomials $p_1, \ldots, p_n \in \mathbf{Poly}$, we claim there is a bijection
$$\Gamma\left(\bigotimes_{i=1}^{n} p_i\right) \cong \prod_{i=1}^{n} \mathbf{Set}\left(\prod_{\substack{1 \le j \le n, \\ j \ne i}} p_j(1), \Gamma(p_i)\right).$$
The $n = 1$ case is clear, and the $n = 2$ case is given by (3.25). Then by induction on n, we have
$$\begin{aligned}\Gamma\left(\bigotimes_{i=1}^{n} p_i\right) &\cong \mathbf{Set}\left(p_n(1), \Gamma\left(\bigotimes_{i=1}^{n-1} p_i\right)\right) \times \mathbf{Set}\left(\prod_{i=1}^{n-1} p_i(1), \Gamma(p_n)\right) && (3.25)\\ &\cong \mathbf{Set}\left(p_n(1), \prod_{i=1}^{n-1} \mathbf{Set}\left(\prod_{\substack{1 \le j \le n-1, \\ j \ne i}} p_j(1), \Gamma(p_i)\right)\right) \times \mathbf{Set}\left(\prod_{i=1}^{n-1} p_i(1), \Gamma(p_n)\right) \\ & && \text{(Inductive hypothesis)} \\ &\cong \prod_{i=1}^{n-1} \mathbf{Set}\left(\prod_{\substack{1 \le j \le n, \\ j \ne i}} p_j(1), \Gamma(p_i)\right) \times \mathbf{Set}\left(\prod_{i=1}^{n-1} p_i(1), \Gamma(p_n)\right), \\ & && \text{(Universal properties of products and internal homs)}\end{aligned}$$
and the result follows.
2. **

Solution to Exercise 3.27.

**

Solution to Exercise 3.29.

Proposition 3.28 follows directly from Proposition 2.38: we have that $\Gamma(0) = \mathbf{Poly}(0, y) \cong 1$ since 0 is initial in **Poly**, and $\Gamma(p + q) = \mathbf{Poly}(p + q, y) \cong \mathbf{Poly}(p + q, y) = \Gamma(p) \times \Gamma(q)$ since $+$ gives coproducts in **Poly**.

Solution to Exercise 3.32.

**

Solution to Exercise 3.34.

**

Solution to Exercise 3.36.

A map of polynomials $\epsilon \colon Sy^S \to 1y^1$ consists of a function $\epsilon_1 \colon S \to 1$, of which there is only one possible, and a function $\epsilon^\sharp \colon S \times 1 \to S$. While $S \times 1$ is not literally equal to S, and hence $\epsilon^\sharp$ can't literally be the identity, there is a canonical isomorphism $S \times 1 \cong S$ given by the functions $(s, 1) \mapsto s$ and $s \mapsto (s, 1)$. It is this canonical isomorphism that is meant here.

Solution to Exercise 3.39.

**

Solution to Exercise 3.41.

**

Solution to Exercise 3.43.

**

Solution to Exercise 3.46.

1. A morphism $HPy^H \otimes Hy^{HP} \to y$ consists of an on-positions function $HPH \to 1$ and an on-directions function $HPH \times 1 \to HHP$. This amounts to a function $HPH \to HHP$. We can easily define this function to be the isomorphism that sends $(h, p, h') \in HPH$ to (h, h', p).
2. To model the way in which you cycle through three possible actions—reaching down and grabbing the chalk, lifting it with your hand, and dropping it—it is simplest to work with a set of 3 possible states. So we will give your dynamics as a polynomial morphism $3y^3 \to HPy^H$, where the yield function $3 \to HP$ indicates what happens at each state, sending $1 \mapsto$ (down, press), $2 \mapsto$ (up, press), and $3 \mapsto$ (up, no press). Then the update function $3H \to 3$ always goes to the next state, regardless of input: it ignores the H coordinate and sends 1 to 2, 2 to 3, and 3 to 1.

Solution to Exercise 3.49.

**

Solution to Exercise 3.50.

**

Solution to Exercise 3.51.

**

Solution to Exercise 3.53.

**

Solution to Exercise 3.58.

**

Solution to Exercise 3.59.

**

Solution to Exercise 3.62.

**

Solution to Exercise 3.69.

We compute $[q, r]$ for various values of $q, r \in \mathbf{Poly}$ using (3.68).

1. If $q := 0$, then $q(1) \cong 0$, so $[q, r]$ is an empty product. Hence $[q, r] \cong 1$.
2. If $q := 1$, then $q(1) \cong 1$ and $q[1] \cong 0$, so $[q, r] \cong r \circ (0y) \cong r(0)$.
3. If $q := y$, then $q(1) \cong 1$ and $q[1] \cong 1$, so $[q, r] \cong r \circ (1y) \cong r$.
4. If $q := A$ for $A \in \mathbf{Set}$, then $q(1) \cong A$ and $q[j] \cong 0$ for every $j \in A$, so $[q, r] \cong \prod_{j \in A}(r \circ (0y)) \cong r(0)^A$.
5. If $q := Ay$ for $A \in \mathbf{Set}$, then $q(1) \cong A$ and $q[j] \cong 1$ for every $j \in A$, so $[q, r] \cong \prod_{j \in A}(r \circ (1y)) \cong r^A$.
6. If $q := y^2 + 2y$ and $r := 2y^3 + 3$, then

$$\begin{aligned}[q, r] &\cong (r \circ (2y))(r \circ (1y))^2 \\ &\cong \left(2(2y)^3 + 3\right)\left(2y^3 + 3\right)^2 \\ &\cong 64y^9 + 204y^6 + 180y^3 + 27\end{aligned}$$

Solution to Exercise 3.70.

We wish to show that for all $p_1, p_2, q \in \mathbf{Poly}$, we have $[p_1 + p_2, q] \cong [p_1, q] \times [p_2, q]$. By (3.68),

$$[p_1 + p_2, q] \cong \left(\prod_{i \in p_1(1)} q \circ (p_1[i]y)\right)\left(\prod_{i \in p_2(1)} q \circ (p_2[i]y)\right) \cong [p_1, q] \times [p_2, q].$$

Solution to Exercise 3.71.

We may compute

$$\begin{aligned}
[q, r] &\cong \prod_{j \in q(1)} r \circ (q[j]y) && (3.68)\\
&\cong \prod_{j \in q(1)} \sum_{k \in r(1)} (q[j]y)^{r[k]} && \text{(Replacing each } y \text{ in } r \text{ by } q[j]y\text{)}\\
&\cong \sum_{f_1 \colon q(1) \to r(1)} \prod_{j \in q(1)} (q[j]y)^{r[f_1(j)]} && (2.24)\\
&\cong \sum_{f_1 \colon q(1) \to r(1)} \left(\prod_{j \in q(1)} q[j]^{r[f_1(j)]}\right)\left(\prod_{j \in q(1)} y^{r[f_1(j)]}\right)\\
&\cong \sum_{f_1 \colon q(1) \to r(1)} \sum_{f^\sharp \in \prod_{j \in q(1)} q[j]^{r[f_1(j)]}} y^{\sum_{j \in q(1)} r[f_1(j)]}\\
&\cong \sum_{f \colon q \to r} y^{\sum_{j \in q(1)} r[f_1(j)]}. && (2.42)
\end{aligned}$$

Solution to Exercise 3.73.

We verify (2.55) as follows:

$$\begin{aligned}
[p, y] \otimes p &\cong \left(\sum_{f \colon p \to y} y^{\sum_{i \in p(1)} y[f_1(i)]}\right) \otimes p && (3.72)\\
&\cong \sum_{f \in \Gamma(p)} y^{p(1)} \otimes \sum_{i \in p(1)} y^{p[i]}\\
&\cong \sum_{f \in \Gamma(p)} \sum_{i \in p(1)} y^{p(1) \times p[i]} && (2.74)\\
&\cong \sum_{f \in \prod_{i \in p(1)} p[i]} \sum_{i \in p(1)} y^{p(1) \times p[i]}. && (3.23)
\end{aligned}$$

Solution to Exercise 3.76.

We have that

$$[y^A, y] \cong \prod_{j \in y^A(1)} y \circ (y^A[j]y) \cong \prod_{j \in 1} Ay \cong Ay,$$

that

$$[Ay, y] \cong \prod_{j \in Ay(1)} y \circ ((Ay)[j]y) \cong \prod_{j \in A} y \cong y^A,$$

and that

$$\begin{aligned}
[p, y] &\cong \sum_{f \colon p \to y} y^{\sum_{i \in p(1)} y[f_1(i)]} && \text{(Exercise 3.71)}\\
&\cong \sum_{f \in \Gamma(p)} y^{\sum_{i \in p(1)} 1}\\
&\cong \Gamma(p) y^{p(1)}.
\end{aligned}$$

Solution to Exercise 3.77.

Given $p \in \textbf{Poly}$ and an isomorphism $[[p, y], y] \cong p$, we wish to show that p is either linear or representable. Applying (3.75) twice, we have that

$$p \cong [[p, y], y] \cong \Gamma\left(\Gamma(p) y^{p(1)}\right) y^{\Gamma(p)}.$$

We can compute the coefficient of p via (3.23) to obtain

$$\Gamma\left(\Gamma(p) y^{p(1)}\right) \cong \prod_{\gamma \in \Gamma(p)} p(1) \cong p(1)^{\Gamma(p)}.$$

Hence

$$p \cong p(1)^{\Gamma(p)} y^{\Gamma(p)}. \tag{3.86}$$

In particular, p is a monomial, so we can write $p \coloneqq By^A$ for some $A, B \in \textbf{Set}$. Then $p(1) \cong B$ and (3.23) tells us that $\Gamma(p) \cong A^B$. It follows from (3.86) that $A \cong A^B$ and that $B \cong B^A$.

We conclude with some elementary set theory. If either one of A or B were 1, then p would be either linear or representable, and we would be done. Meanwhile, if either one of A or B were 0, then the other would be 1, and we would again be done. Otherwise, $|A|, |B| \geq 2$. But by Cantor's theorem,

$$|B| < \left|2^B\right| \leq \left|A^B\right| = |A| \qquad \text{and} \qquad |A| < \left|2^A\right| \leq \left|B^A\right| = |B|,$$

a contradiction.

Solution to Exercise 3.80.

The isomorphism $\textbf{Poly}(p, q) \cong [p, q](1)$ follows directly from Exercise 3.71 when both sides are applied to 1. Alternatively, we can apply (3.79). Since $p \cong y \otimes p$, we have that

$$\begin{aligned} \textbf{Poly}(p, q) &\cong \textbf{Poly}(y \otimes p, q) \\ &\cong \textbf{Poly}(y, [p, q]) && (3.79) \\ &\cong [p, q](1). && \text{(Yoneda lemma)} \end{aligned}$$

Solution to Exercise 3.82.

1. To obtain the evaluation map $\text{eval}\colon [p, q] \otimes p \longrightarrow q$, we consider the following special case of the isomorphism (3.79):
$$\textbf{Poly}([p, q] \otimes p, q) \cong \textbf{Poly}([p, q], [p, q]).$$
Then the evaluation map is the map corresponding to the identity morphism on $[p, q]$ under the above isomorphism. To recover this map, we can start from the identity morphism on $[p, q]$ and work our way along a chain of natural isomorphisms from $\textbf{Poly}([p, q], [p, q])$ until we get to $\textbf{Poly}([p, q] \otimes p, q)$. To start, Exercise 3.71 implies that
$$\begin{aligned} \textbf{Poly}([p, q], [p, q]) &\cong \textbf{Poly}\left(\sum_{f\colon p \to q} \prod_{i' \in p(1)} y^{q[f_1(i')]}, \prod_{i \in p(1)} \sum_{j \in q(1)} (p[i]y)^{q[j]}\right) \\ &\cong \prod_{f\colon p \to q} \prod_{i \in p(1)} \textbf{Poly}\left(\prod_{i' \in p(1)} y^{q[f_1(i')]}, \sum_{j \in q(1)} (p[i]y)^{q[j]}\right), \end{aligned}$$
where the second isomorphism follows from the universal properties of products and coproducts. In particular, under this isomorphism, the identity morphism on $[p, q]$ corresponds to a collection of morphisms, namely for each $f\colon p \to q$ and each $i \in p(1)$ the composite
$$\prod_{i' \in p(1)} y^{q[f_1(i')]} \to y^{q[f_1(i)]} \to \sum_{g\colon q[f_1(i)] \to p[i]} y^{q[f_1(i)]} \cong (p[i]y)^{q[f_1(i)]} \to \sum_{j \in q(1)} (p[i]y)^{q[j]}$$

of the canonical projection with index $i' = i$, the canonical inclusion with index $g = f_i^\sharp$, and the canonical inclusion with index $j = f_1(i)$. On positions, this map picks out the position of $\sum_{j \in q(1)} (p[i]y)^{q[j]}$ corresponding to $j = f_1(i) \in q(1)$ and $f_i^\sharp \colon q[f_1(i)] \to p[i]$; on directions, the map is the canonical inclusion $q[f_1(i)] \to \sum_{i' \in p(1)} q[f_1(i')]$ with index $i' = i$.

We can reinterpret each of these maps as a map

$$y^{p[i] \times \sum_{i' \in p(1)} q[f_1(i')]} \to \sum_{j \in q(1)} y^{q[j]} \cong q$$

that, on positions, picks out the position $f_1(i) \in q(1)$ of q and, on directions, is the map $q[f_1(i)] \to p[i] \times \sum_{i' \in p(1)} q[f_1(i')]$ induced by the universal property of products applied to the map $f_i^\sharp \colon q[f_1(i)] \to p[i]$ and the inclusion $q[f_1(i)] \to \sum_{i' \in p(1)} q[f_1(i')]$. Then by the universal property of coproducts, this collection of maps induces a single map $\mathrm{eval} \colon [p, q] \otimes p \to q$ that sends each position $f \colon p \to q$ of $[p, q]$ and position $i \in p(1)$ of p to the position $f_1(i)$ of q, with the same behavior on directions as the corresponding map described previously.

2. **

Solution to Exercise 3.83.

1. Given a set S, we wish to obtain the map $Sy^S \to y$ whose on-diretions map is the identity by using eval and Example 3.74. The example shows that

$$[Sy, y] \otimes (Sy) \cong y^S \otimes (Sy) \cong Sy^S,$$

so by setting $p := Sy$ and $q := S$ in (3.81), we obtain an evaluation map $\mathrm{eval} \colon Sy^S \to y$. By the solution to Exercise 3.82 #1, given a position $s \in S$ of Sy^S, the evaluation map on directions is the map $1 \to S$ that picks out s. In other words, it is indeed the identity on directions.

2. **

Chapter 4

More categorical properties of polynomials

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 Part II if so inclined. Better yet might be to just gently leaf through Chapter 4, to see how well-behaved and versatile the category **Poly** really is.

4.1 Special polynomials and adjunctions

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

a) constant polynomials $0, 1, 2, A$;
b) linear polynomials $0, y, 2y, Ay$;
c) pure-power (or representable) polynomials $1, y, y^2, y^A$; and
d) 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 4.2 and 4.3. The third puts a copy of $\mathbf{Set}^{\mathrm{op}}$ inside **Poly**. Finally, the fourth puts a copy of bimorphic lenses inside **Poly**, as we saw in Section 2.4.4.

Exercise 4.1 (Solution here). Which of the four classes above are closed under

1. the cocartesian monoidal structure $(0, +)$ (i.e. addition)?
2. the cartesian monoidal structure $(1, \times)$ (i.e. multiplication)?
3. the parallel monoidal structure $(y, \otimes)$ (i.e. taking the parallel product)?
4. composition of polynomials $p \circ q$? ◊

Proposition 4.2. There is a fully faithful functor $\mathbf{Set} \to \mathbf{Poly}$ sending $A \mapsto Ay^0 = A$.

Proof. By (2.43), a map $f\colon Ay^0 \to By^0$ consists of a function $f\colon A \to B$ and, for each $a \in A$, a function $0 \to 0$. There is only one function $0 \to 0$, so f can be identified with just a map of sets $A \to B$. □

Proposition 4.3. There is a fully faithful functor **Set** → **Poly** sending $A \mapsto Ay$.

Proof. By (2.43), a map $f\colon Ay^1 \to By^1$ consists of a function $f\colon A \to B$ and for each $a \in A$ a function $1 \to 1$. There is only one function $1 \to 1$, so f can be identified with just a map of sets $A \to B$. □

Theorem 4.4. **Poly** has an adjoint quadruple with **Set**:

$$\begin{tikzcd}[column sep=huge]
\mathbf{Set} \arrow[r, leftarrow, shift left=9, "p(0)"] \arrow[r, phantom, shift left=6, "\Rightarrow"] \arrow[r, shift left=3, "A" description] \arrow[r, phantom, "\Leftarrow"] \arrow[r, leftarrow, shift right=3, "p(1)" description] \arrow[r, phantom, shift right=6, "\Rightarrow"] \arrow[r, shift right=9, "Ay"'] & \mathbf{Poly}
\end{tikzcd} \tag{4.5}$$

where the functors have been labeled by where they send $A \in$ **Set** and $p \in$ **Poly**.
Both rightward functors are fully faithful.

Proof. For any set A, there is a functor **Poly** → **Set** given by sending p to $p(A)$; by the Yoneda lemma, it is the functor $\mathbf{Poly}(y^A, -)$. This, together with Propositions 4.2 and 4.3, gives us the four functors and the fact that the two rightward functors are fully faithful. It remains to provide the following three natural isomorphisms:

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

All three come from our formula (2.42) for computing general hom-sets in **Poly**; we leave the details to the reader in Exercise 4.6. □

Exercise 4.6 (Solution here). Here we prove the remainder of Theorem 4.4 using (2.42):

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

Exercise 4.7 (Solution here). 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 4.4 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 4.10. However, the fact that $p \mapsto p(1)$ is itself the left adjoint of the left adjoint of $p \mapsto p(0)$—and hence that we have the *quadruple* of adjunctions in (4.5)—is special to $A = 0, 1$.

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

Proposition 4.8. There is a two-variable adjunction between **Poly**, **Set**, and **Poly**:

$$\mathbf{Poly}(Ap, q) \cong \mathbf{Set}(A, \mathbf{Poly}(p, q)) \cong \mathbf{Poly}(p, q^A). \tag{4.9}$$

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

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

The middle set is naturally isomorphic to $\mathbf{Set}(A, \mathbf{Poly}(p, q))$, completing the proof. □

Replacing p with y^B in (4.9), we obtain the following using the Yoneda lemma.

Corollary 4.10. For any set B there is an adjunction

$$\begin{tikzcd} \mathbf{Set} \arrow[r, shift left, "Ay^B"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift left, "q(B)"] \end{tikzcd}$$

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

Exercise 4.11 (Solution here). Prove Corollary 4.10 from Proposition 4.8. ◊

Proposition 4.12. The Yoneda embedding $A \mapsto y^A$ has a left adjoint

$$\begin{tikzcd} \mathbf{Set}^{\mathrm{op}} \arrow[r, shift left, "y^-"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left, "\Gamma"] \end{tikzcd}$$

where $\Gamma(p) := \mathbf{Poly}(p, y) \cong \prod_{i \in p(1)} p[i]$, as in (3.22) and (3.23). That is, there is a natural isomorphism

$$\mathbf{Poly}(p, y^A) \cong \mathbf{Set}(A, \Gamma(p)). \tag{4.13}$$

Proof. By (2.42), we have the natural isomorphism

$$\mathbf{Poly}(p, y^A) \cong \prod_{i \in p(1)} p[i]^A,$$

which in turn is naturally isomorphic to $\mathbf{Set}(A, \Gamma(p))$ by (3.23). □

Corollary 4.14 (Principle monomial)**.** There is an adjunction

$$\begin{tikzcd}\mathbf{Poly} \arrow[r, shift left, "{(p(1),\Gamma(p))}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Set}\times\mathbf{Set}^{\mathrm{op}} \arrow[l, shift left, "Ay^B"]\end{tikzcd}$$

where the functors are labeled by where they send $p \in \mathbf{Poly}$ and $(A, B) \in \mathbf{Set}\times\mathbf{Set}^{\mathrm{op}}$. That is, there is a natural isomorphism

$$\mathbf{Poly}(p, Ay^B) \cong \mathbf{Set}(p(1), A) \times \mathbf{Set}(B, \Gamma(p)). \tag{4.15}$$

Proof. By the universal property of the product of A and y^B, we have a natural isomorphism

$$\mathbf{Poly}(p, Ay^B) \cong \mathbf{Poly}(p, A) \times \mathbf{Poly}(p, y^B).$$

Then the desired natural isomorphism follows from Exercise 4.6 #2 and (4.13). □

Exercise 4.16 (Solution here). Use (4.15) together with (3.75) and (3.79) to find an alternative proof for Proposition 3.24, i.e. that there is an isomorphism

$$\Gamma(p \otimes q) \cong \mathbf{Set}\big(q(1), \Gamma(p)\big) \times \mathbf{Set}\big(p(1), \Gamma(q)\big).$$

for any $p, q \in \mathbf{Poly}$. ◊

4.2 Epi-mono factorization

Proposition 4.17. Let $f\colon p \to q$ be a morphism in **Poly**. It is a monomorphism if and only if the on-positions function $f_1\colon p(1) \to q(1)$ is a monomorphism in **Set** and, for each $i \in p(1)$, the on-directions function $f_i^\sharp\colon q[f_1(i)] \to p[i]$ is an epimorphism in **Set**.

Proof. To prove the forward direction, suppose that f is a monomorphism. Since $p \mapsto p(1)$ is a right adjoint (Theorem 4.4), it preserves monomorphisms, so the on-positions function f_1 is also a monomorphism.

We now need to show that for any $i \in p(1)$, the on-directions function $f_i^\sharp\colon q[f_1(i)] \to p[i]$ is an epimorphism. Suppose we are given a set A and a pair of functions $g^\sharp, h^\sharp\colon p[i] \rightrightarrows A$ with $f_i^\sharp \mathbin{⨟} g^\sharp = f_i^\sharp \mathbin{⨟} h^\sharp$. Then there exist morphisms $g, h\colon y^A \rightrightarrows p$ whose on-positions functions both pick out i and whose on-directions functions are $g^\sharp$ and $h^\sharp$, so that $g \mathbin{⨟} f = h \mathbin{⨟} f$. As f is a monomorphism, $g = h$; in particular, their on-directions functions $g^\sharp$ and $h^\sharp$ are equal, as desired.

Conversely, 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 and $g, h\colon r \rightrightarrows p$ be two morphisms such that $g \mathbin{⨟} f = h \mathbin{⨟} f$. Then $g_1 \mathbin{⨟} f_1 = h_1 \mathbin{⨟} f_1$, which implies $g_1 = h_1$; we'll consider g_1

the default representation. We also have that $f^\sharp_{g_1(k)} ⨟ g^\sharp_k = f^\sharp_{g_1(k)} ⨟ h^\sharp_k$ for any $k \in r(1)$. But $f^\sharp_{g_1(k)}$ is an epimorphism, so in fact $g^\sharp_k = h^\sharp_k$, as desired. □

Example 4.18. Choose a finite nonempty set k for $1 \leq k \in \mathbb{N}$, e.g. $\mathsf{k} = \mathsf{12}$. There is a monomorphism

$$f \colon \mathsf{k}y^{\mathsf{k}} \to \mathbb{N}y^{\mathbb{N}}$$

such that the trajectory "going around and around the k-clock" comes from the usual counting trajectory Example 3.3 $\mathbb{N}y^{\mathbb{N}} \to y$.

On positions, we have $f_1(i) = i$ for all $i \in \mathsf{k}$. On directions, for any $i \in \mathsf{k}$, we have $f^\sharp_i(n) = n \mod k$ for all $n \in \mathbb{N}$.

Exercise 4.19 (Solution here). In Example 4.18, we gave a map $\mathsf{12}y^{\mathsf{12}} \to \mathbb{N}y^{\mathbb{N}}$. 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 $\mathsf{12}y^{\mathsf{12}} \to p$ would be seen to relate to the behavior of the old system $\mathbb{N}y^{\mathbb{N}} \to p$. ◊

Proposition 4.20. Let $f \colon p \to q$ be a morphism in **Poly**. It is an epimorphism if and only if 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

$$f^\flat_j \colon q[j] \to \prod_{\substack{i \in p(1), \\ f_1(i) = j}} p[i]$$

from (2.50) is a monomorphism.

Proof. To prove the forward direction, suppose that f is an epimorphism. Since $p \mapsto p(1)$ is a left adjoint (Theorem 4.4), it preserves epimorphisms, so the on-positions function f_1 is also a epimorphism.

We now need to show that for any $j \in q(1)$, the induced function $f^\flat_j$ is a monomorphism. Suppose we are given a set A and a pair of functions $g', h' \colon A \rightrightarrows q[j]$ with $g' ⨟ f^\flat_j = h' ⨟ f^\flat_j$. 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.

Then we can construct morphisms $g, h \colon q \rightrightarrows y^A + 1$ whose on-positions functions both send j to the first position, corresponding to y^A, and all other positions to the second position, corresponding to 1. In addition, we let the on-directions functions be $g^\sharp_j \coloneqq g'$ and $h^\sharp_j \coloneqq h'$. Then $f ⨟ g = f ⨟ h$. As f is an epimorphism, $g = h$; in particular, their on-directions functions are equal, so $g' = h'$, as desired.

Conversely, 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 and $g, h \colon q \rightrightarrows r$ be two morphisms such that $f \mathbin{⨾} g = f \mathbin{⨾} h$. Then $f_1 \mathbin{⨾} g_1 = f_1 \mathbin{⨾} h_1$, which implies $g_1 = h_1$; we'll consider g_1 the default representation. We also have that $g^\sharp_{f_1(i)} \mathbin{⨾} f_i^\sharp = h^\sharp_{f_1(i)} \mathbin{⨾} f_i^\sharp$ for any $i \in p(1)$. It follows that, for any $j \in q(1)$, the two composites

$$\begin{tikzcd} r[g_1(j)] \arrow[r, shift left, "g_j^\sharp"] \arrow[r, shift right, "h_j^\sharp"'] & q[j] \arrow[r, "f_j^\flat"] & \prod_{\substack{i \in p(1),\\ f_1(i) = j}} p[i] \end{tikzcd}$$

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

Exercise 4.21 (Solution here). Show that the only way for a map $p \to y$ to *not* be an epimorphism is when $p = 0$. ◊

Exercise 4.22 (Solution here). Let A and B be sets and AB their product. Find an epimorphism $y^A + y^B \twoheadrightarrow y^{AB}$. ◊

Exercise 4.23 (Solution here). Suppose a polynomial morphism is both a monomorphism and an epimorphism; it is then an isomorphism? (That is, is **Poly** *balanced*?)

Hint: You may use the following facts.

1. A function that is both a monomorphism and an epimorphism in **Set** is an isomorphism.
2. A polynomial morphism is an isomorphism if and only if the on-positions function is an isomorphism and every on-directions function is an isomorphism.

◊

Exercise 4.24 (Epi-mono factorization; solution here).

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

4.3 Limits, colimits, and cartesian closure

We have already seen that **Poly** has all coproducts (Proposition 2.38) and products (Proposition 2.65). We will now see that **Poly** has all limits and colimits, and moreover it is cartesian closed.

4.3.1 Cartesian closure

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

$$r^q := \prod_{j \in q(1)} r \circ (y + q[j]) \tag{4.25}$$

where $\circ$ denotes composition.

Before proving that this really is an exponential in **Poly**, which we do in Theorem 4.28, we first get some practice with it.

Example 4.26. Let A be a set. We've been writing the polynomial Ay^0 simply as A, so it better be true that the there is an isomorphism

$$y^A \cong y^{Ay^0}$$

in order for the notation to be consistent. Luckily, this is true. By (4.25), we have

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

Exercise 4.27 (Solution here). Compute the following exponentials in **Poly** using (4.25):

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 $A, B \in$ **Set**. ◊

Theorem 4.28. 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 (4.25).

Proof. We have the following chain of natural isomorphisms:

$$\begin{aligned}\mathbf{Poly}(p, r^q) &\cong \mathbf{Poly}\Big(p, \prod_{j \in q(1)} r \circ (y + q[j])\Big) && (4.25)\\ &\cong \prod_{i \in p(1)} \prod_{j \in q(1)} \mathbf{Poly}\big(y^{p[i]}, r \circ (y + q[j])\big)\end{aligned}$$

(Universal property of (co)products)

$$
\begin{aligned}
&\cong \prod_{i\in p(1)} \prod_{j\in q(1)} r \circ (p[i] + q[j]) && \text{(Yoneda lemma)}\\
&\cong \prod_{i\in p(1)} \prod_{j\in q(1)} \sum_{k\in r(1)} (p[i] + q[j])^{r[k]} \\
&\cong \prod_{(i,j)\in(p\times q)(1)} \sum_{k\in r(1)} (p\times q)[(i,j)]^{r[k]} && (2.69)\\
&\cong \textbf{Poly}(p\times q, r). && (2.42)
\end{aligned}
$$

□

Exercise 4.29 (Solution here). Use Theorem 4.28 to show that for any polynomials p, q, there is a canonical evaluation map

$$\text{eval}\colon p^q \times q \to p.$$

◊

4.3.2 Limits and colimits

Theorem 4.30. The category **Poly** has all limits.

Proof. A category has all limits if and only if it has products and equalizers, so by Proposition 2.65, it suffices to show that **Poly** has equalizers.

We claim that equalizers in **Poly** are simply equalizers on positions and coequalizers on directions. More precisely, let $f, g\colon p \rightrightarrows q$ be two maps of polynomials. We construct the equalizer p' of f and g as follows.[1] We define its set of positions $p'(1)$ to be the equalizer of $f_1, g_1\colon p(1) \rightrightarrows q(1)$ in **Set**; that is,

$$p'(1) \coloneqq \{i \in p(1) \mid f_1(i) = g_1(i)\}.$$

Then for each $i \in p'(1)$, we can define the set of directions $p'[i]$ to be the coequalizer of $f_i^\sharp, g_i^\sharp\colon q[f_1(i)] \rightrightarrows p[i]$. In this way, we obtain a polynomial p' that comes equipped with a morphism $e\colon p' \to p$. One can check that p' together with e satisfies the universal property of the equalizer of f and g; see Exercise 4.31. □

Exercise 4.31 (Solution here). Complete the proof of Theorem 4.30 as follows:

1. We said that p' comes equipped with a morphism $e\colon p' \to p$; what is it?
2. Show that $e \mathbin{⨟} f = e \mathbin{⨟} g$.
3. Show that e is the equalizer of the pair f, g. ◊

---

[1] If we're being precise, a "(co)equalizer" is an object equipped with a map, but we will use the term to refer to either just the object or just the map when the context is clear.

Example 4.32 (Pullbacks in **Poly**). Given $q, q', r \in \mathbf{Poly}$ and morphisms $q \xrightarrow{f} r \xleftarrow{f'} q'$, the pullback

$$\begin{tikzcd}
p \arrow[r, "g'"] \arrow[d, "g"'] \arrow[dr, phantom, "\lrcorner", very near start] & q' \arrow[d, "f'"] \\
q \arrow[r, "f"'] & r
\end{tikzcd}$$

is given as follows. The set of positions of p is the pullback of the positions of q and q' over those of r in **Set**. Then at each position $(i, i') \in p(1) \subseteq q(1) \times q'(1)$ with $f_1(i) = f'_1(i')$, we take the set of directions $p[(i, i')]$ to be the pushout of the directions $q[i]$ and $q'[i']$ over $r[f_1(i)] = r[f'_1(i')]$ in **Set**. These pullback and pushout squares also given the morphisms g and g' on positions and on directions:

$$\begin{tikzcd}
p(1) \arrow[r, "g'_1"] \arrow[d, "g_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & q'(1) \arrow[d, "f'_1"] \\
q(1) \arrow[r, "f_1"'] & r(1)
\end{tikzcd}
\qquad \text{and} \qquad
\begin{tikzcd}
p[(i, i')] \arrow[dr, phantom, "\lrcorner", very near start] & q'[i'] \arrow[l, "(g')^\sharp_{(i,i')}"'] \\
q[i] \arrow[u, "g^\sharp_{(i,i')}"] & r[f_1(i)] \arrow[l, "f^\sharp_i"] \arrow[u, "(f')^\sharp_{i'}"']
\end{tikzcd}
\tag{4.33}$$

Exercise 4.34 (Solution here). Let p be any polynomial.

1. There is a canonical choice of morphism $\eta \colon p \to p(1)$; what is it?
2. Given an element $i \in p(1)$, i.e. a function (or morphism between constant polynomials) $i \colon 1 \to p(1)$, let p_i be the pullback

$$\begin{tikzcd}
p_i \arrow[r, "g"] \arrow[d, "f"'] \arrow[dr, phantom, "\lrcorner", very near start] & p \arrow[d, "\eta"] \\
1 \arrow[r, "i"'] & p(1)
\end{tikzcd}$$

   What is p_i? What are the maps $f \colon p_i \to 1$ and $g \colon p_i \to p$? ◊

Exercise 4.35 (Solution here). Let $q := y^2 + y$, $q' := 2y^3 + y^2$, and $r := y + 1$.

1. Choose morphisms $f \colon q \to r$ and $f' \colon q' \to r$ and write them down.
2. Find the pullback of $q \xrightarrow{f} r \xleftarrow{f'} q'$. ◊

Theorem 4.36. The category **Poly** has all colimits.

Proof. A category has all colimits if and only if it has coproducts and coequalizers, so by Proposition 2.38, it suffices to show that **Poly** has coequalizers.

Let $s, t\colon p \rightrightarrows q$ be two maps of polynomials. We construct the coequalizer q' of s and t as follows. The pair of functions $s_1, t_1\colon p(1) \rightrightarrows q(1)$ define a graph $G\colon \boxed{\bullet \rightrightarrows \bullet} \to \mathbf{Set}$ with vertices in $q(1)$, edges in $p(1)$, sources indicated by s_1, and targets indicated by t_1. Then the set C of connected components of G is given by the coequalizer $g_1\colon q(1) \to C$ of s_1 and t_1. We define the set of positions of q' to be C. Each set of directions of q' will be a limit of a diagram of sets of directions of 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 $V_c \coloneqq g_1^{-1}(c)$ and edges $E_c \coloneqq s_1^{-1}(g_1^{-1}(c)) = t_1^{-1}(g_1^{-1}(c))$. 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 set of directions $p[e]$ (resp. $q[v]$).

The category of elements $\int G_c$ has objects $E_c + V_c$ and two kinds of (non-identity) morphisms, $e \to s_1(e)$ and $e \to t_1(e)$, associated to each $e \in E_c$, all pointing from an object in E_c to an object in V_c. There is a functor $F\colon (\int G_c)^{\text{op}} \to \mathbf{Set}$ sending every $v \mapsto q[v]$, every $e \mapsto p[e]$, and every morphism to a function between them, namely either $s_e^\sharp\colon q[s_1(e)] \to p[e]$ or $t_e^\sharp\colon q[t_1(e)] \to p[e]$. So we can define $q'[c]$ to be the limit of F in $\mathbf{Set}$.

We claim that $q' \coloneqq \sum_{c \in C} y^{q'[c]}$ is the coequalizer of s and t. We leave the complete proof to the interested reader in Exercise 4.37. □

Exercise 4.37 (Solution here). Complete the proof of Theorem 4.36 as follows:

1. Provide a map $g\colon q \to q'$.
2. Show that $s \mathbin{⨾} g = t \mathbin{⨾} g$.
3. Show that g is a coequalizer of the pair s, t. ◊

Example 4.38. Given a diagram in **Poly**, one could either take its colimit as a diagram of *polynomial* functors (i.e. its colimit in **Poly**) or its colimit simply as a diagram of functors (i.e. its colimit in $\mathbf{Set}^{\mathbf{Set}}$). These can yield different results, as evidenced by the fact that, per the co-Yoneda lemma, *every* functor $\mathbf{Set} \to \mathbf{Set}$—even those that are not polynomials—can be written as the colimit of representable functors in $\mathbf{Set}^{\mathbf{Set}}$, yet the colimit of the same representables in **Poly** can only be another polynomial.

As a concrete example, consider the two projections $y^2 \to y$, which form the diagram

$$y^2 \rightrightarrows y. \tag{4.39}$$

According to Theorem 4.36, the colimit of (4.39) in **Poly** has the coequalizer of $1 \rightrightarrows 1$, namely 1, as its set of positions, and the limit of the diagram $1 \rightrightarrows 2$ consisting of the two inclusions as its sole set of directions. But this latter limit is just 0, so in fact the colimit of (4.39) in **Poly** is 1.

But as functors, the colimit of (4.39) is the functor

$$X \mapsto \begin{cases} 0 & \text{if } X = 0 \\ 1 & \text{if } X \neq 0 \end{cases}$$

Exercise 4.40 (Solution here). By Proposition 1.21, for any polynomial p, there are canonical maps

$$\epsilon \colon p(1)y \to p \qquad \text{and} \qquad \eta \colon p \to y^{\Gamma(p)}.$$

1. Characterize the behavior of the canonical map $\epsilon \colon p(1)y \to p$.
2. Characterize the behavior of the canonical map $\eta \colon p \to y^{\Gamma(p)}$.
3. Show that the following is a pushout in **Poly**:

$$\begin{tikzcd}
p(1)y \arrow[r, "!"] \arrow[d, "\epsilon"'] & y \arrow[d, "!"] \\
p \arrow[r, "\eta"'] \arrow[ur, phantom, "\ulcorner", very near end] & y^{\Gamma(p)}
\end{tikzcd} \tag{4.41}$$

◊

Proposition 4.42. For polynomials p, q, the following is a pushout:

$$\begin{tikzcd}
p(1)y \otimes q(1)y \arrow[r] \arrow[d] & p(1)y \otimes q \arrow[d] \\
p \otimes q(1)y \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & p \otimes q
\end{tikzcd}$$

Proof. All the maps shown are identities on positions, so the displayed diagram is the coproduct over all $(i, j) \in p(1) \times q(1)$ of the diagram shown left

$$\begin{tikzcd}
y \arrow[r] \arrow[d] & y^{q[j]} \arrow[d] & 1 & q[j] \arrow[l] \\
y^{p[i]} \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & y^{p[i] \times q[j]} & p[i] \arrow[u] & p[i] \times q[j] \arrow[l] \arrow[u] \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

where we used $(p \otimes q)[(i, j)] \cong p[i] \times q[j]$. This is the image under the Yoneda embedding of the diagram of sets shown right, which is clearly a pullback. The result follows by Proposition 4.12. □

This means that to give a map $\varphi \colon p \otimes q \to r$, it suffices to give two maps $\varphi_p \colon p \otimes q(1)y \to r$ and $\varphi_q \colon p(1)y \otimes q \to r$ that agree on positions. The map φ_p says how information about q's position is transferred to p, and the map φ_q says how information about p's position is transferred to q.

Corollary 4.43. Suppose given polynomials $p_1, \ldots, p_n \in \mathbf{Poly}$. Then $p_1 \otimes \cdots \otimes p_n$ is isomorphic to the wide pushout

$$\operatorname{colim}\left(\begin{tikzcd} & p_1(1)y \otimes \cdots \otimes p_n(1)y \arrow[dl] \arrow[dr] & \\ p_1 \otimes p_2(1)y \otimes \cdots \otimes p_n(1)y & \cdots & p_1(1)y \otimes \cdots \otimes p_{n-1}(1)y \otimes p_n \end{tikzcd}\right)$$

Proof. We proceed by induction on $n \in \mathbb{N}$. When $n = 0$, the wide pushout has no legs and the empty tensor product is y, so the result holds. If the result holds for n, then it holds for $n + 1$ by Proposition 4.42. □

We can use Corollary 4.43 to characterize interaction patterns. Given polynomials $p_1, \ldots, p_n$, and q, and a map

$$\varphi \colon p_1 \otimes \cdots \otimes p_n \to q$$

we may want to know which p_i's can "hear" which p_j's. To make this precise, we will associate to φ a simple directed graph that describes the information flow. Given a set V, define

$$\mathbf{SimpleGraph}(V) \coloneqq \{A \subseteq V \times V \mid \forall(v \in V).(v, v) \notin A\}.$$

In fact, we will start with a more general result that explains how maps such as φ can be *mode-dependent*. But before we do, let's explain the idea in an example.

Example 4.44 (Introducing random process). What is it about flipping a coin or rolling a die that is random? This is a different question than whether the coin or die is fair or even consistent. What's important is that we don't have any influence over the result of the outcome.

Suppose that we have a system $\varphi \colon p \otimes c \to r$, where we think of p as a player and c as a coin. By Proposition 4.42 we know that φ can be identified with a commuting diagram

$$\begin{tikzcd} p(1)y \otimes c(1)y \arrow[r] \arrow[d] & p(1)y \otimes c \arrow[d] \\ p \otimes c(1)y \arrow[r] & r \end{tikzcd}$$

Assuming for simplicity that the player and coin form a closed system, i.e. that$r = y$, the map φ can be identified with two maps: $\varphi_c \colon p(1)y \otimes c \to y$ and $\varphi_p \colon p \otimes c(1)y \to y$. To say that p's position does not influence c is to say that the former map factors through the projection $\pi \colon p(1)y \otimes c \to c$

$$p(1)y \otimes c \xrightarrow{\pi} c \to y.$$

In general, we will be interested in the interaction pattern between a number of systems, in terms of who influences who.

Example 4.45. Carrying on from Example 4.44, suppose given polynomials $p_1, \ldots, p_n, r$ and a map

$$\varphi\colon p_1 \otimes \cdots \otimes p_n \to r.$$

By Corollary 4.43, φ can be identified with a tuple of maps

$$\varphi_i\colon p_i \otimes \bigotimes_{j \neq i} p_j(1)y \longrightarrow r$$

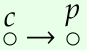
that agree on positions $\varphi(1)\colon p(1) \otimes \cdots \otimes p_n(1) \to r(1)$.

Now for each $1 \leq i \leq n$, we can ask for the smallest subset $A_i \subseteq \{j \mid 1 \leq j \leq n, j \neq i\}$ such that φ_i factors through the projection

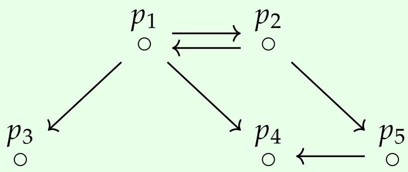
$$p_i \otimes \bigotimes_{j \neq i} p_j(1)y \xrightarrow{\pi_A} p_i \otimes \bigotimes_{j \in A_i} p_j(1)y \to r.$$

The result is a simple directed graph: its vertices are the numbers $V \coloneqq \{i \mid 1 \leq i \leq n\}$ and for each vertex we have a subset $A_i \subseteq V \setminus \{i\}$, which we consider as the arrows with target i. The set of all arrows in this graph is $A \coloneqq \sum_i A_i$.

The simple directed graph associated to the player-and-coin model in Example 4.44 would be



indicating that the coin's position can be noticed by the player but the player's position cannot be noticed by the coin. In general, we may have something like the following:



which says that p_1 is heard by p_2, p_3, p_4 but not by p_5, etc.

For any $p \in \mathbf{Poly}$, let $p/\mathbf{Poly}$ denote the coslice category, i.e. the category whose objects are maps $p \to q$ emanating from p, and whose morphisms are commutative triangles.

Proposition 4.46 (Interaction graphs)**.** For any polynomials $p_1, \ldots, p_n \in \mathbf{Poly}$ there is an adjunction

$$\begin{tikzcd}\mathbf{SimpleGraph}(\mathsf{n})^{\mathrm{op}} \arrow[r, shift left=1.5] \arrow[r, phantom, "\Rightarrow"] & p_1 \otimes \cdots \otimes p_n / \mathbf{Poly} \arrow[l, shift left=1.5]\end{tikzcd}$$

Moreover, the left adjoint is fully faithful if and only if $n \leq 1$ or $|p_i(1)| \geq 2$ for each $1 \leq i \leq n$, i.e. if and only if each polynomial has at least two positions.

Proof. ** □

4.4 Monoidal $*$-bifibration over Set

We will see that the functor $p \mapsto p(1)$ has special properties making it what [Shu08] 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 $f\colon A \to p(1)$, which we can also think of as a morphism between constant polynomials. Then we get a new polynomial f^*p with positions A, as follows. It is given by a pullback

$$\begin{tikzcd}f^*p \arrow[r, "\mathrm{cart}"] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & p \arrow[d, "\eta_p"] \\ A \arrow[r, "f"'] & p(1)\end{tikzcd} \tag{4.47}$$

Here η_p is the unit of the adjunction $\begin{tikzcd}\mathbf{Set} \arrow[r, shift left=1.5, "A"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly} \arrow[l, shift left=1.5, "p(1)"]\end{tikzcd}$. By Example 4.32, this pullback yields an on-positions pullback

$$\begin{tikzcd}(f^*p)(1) \arrow[r, "\mathrm{cart}_1"] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & p(1) \arrow[d, "(\eta_p)_1"] \\ A \arrow[r, "f"'] & p(1)\end{tikzcd} \tag{4.48}$$

in **Set**. Since η_p is an isomorphism on positions, the map $f^*p \to A$ is also an isomorphism on positions and $\mathrm{cart}_1 = f$.

Example 4.32 also implies that for each position $a \in A$, the pullback in (4.47) yields an on-directions pushout

$$\begin{tikzcd}(f^*p)[a] \arrow[dr, phantom, very near end, "\lrcorner"] & p[f(a)] \arrow[l, "\mathrm{cart}^\sharp_a"'] \\ 0 \arrow[u] & 0 \arrow[l, "f^\sharp_a"] \arrow[u, "(\eta_p)^\sharp_i"']\end{tikzcd} \tag{4.49}$$

in **Set**. The on-directions function $f_a^\sharp \colon 0 \to 0$ is trivially an isomorphism, so each on-directions function $\text{cart}_a^\sharp$ is an isomorphism, too. Hence

$$f^* p \cong \sum_{a \in A} y^{p[f(a)]}.$$

We'll see this as part of a bigger picture in Proposition 4.64 and Theorem 4.65, but first, we introduce vertical and cartesian morphisms in **Poly** (thus explaining why we named our morphism "cart").

Definition 4.50 (Vertical, cartesian)**.** Let $f \colon p \to q$ be a morphism of polynomials. It is called *vertical* if $f_1 \colon p(1) \to q(1)$ is an isomorphism. It is called *cartesian* if, for each $i \in p(1)$, the function $f_i^\sharp \colon q[f(i)] \to p[i]$ is an isomorphism.

Proposition 4.51. Every morphism in **Poly** can be uniquely factored as a vertical morphism followed by a cartesian morphism.

Proof. Recall from (2.43) that a morphism in **Poly** can be written as to the left; we can thus rewrite it as to the right:

$$\begin{tikzcd}
p(1) \arrow[rr, "f_1"] \arrow[dr, bend right, "{p[-]}"'] & {} \arrow[d, phantom, "\overset{f^\sharp}{\Leftarrow}"] & q(1) \arrow[dl, bend left, "{q[-]}"] \\
& \mathbf{Set} &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
p(1) \arrow[rr, equal] \arrow[drr, bend right, "{p[-]}"'] & & p(1) \arrow[rr, "f_1"] \arrow[d, dashed, "{q[f_1(-)]}"] & & q(1) \arrow[dll, bend left, "{q[-]}"] \\
& {} \arrow[r, phantom, "\overset{f^\sharp}{\Leftarrow}"] & \mathbf{Set} & &
\end{tikzcd}$$

The intermediary object $\sum_{i \in p(1)} y^{q[f_1(i)]}$ is clearly unique up to isomorphism. □

Proposition 4.52. Vertical morphisms satisfy 2-out-of-3: given $p \xrightarrow{f} q \xrightarrow{g} r$ with $h = f \mathbin{⨟} g$, if any two of f, g, h are vertical, then so is the third.

If g is cartesian, then h is cartesian if and only if f is cartesian.

Proof. Given $h = f \mathbin{⨟} g$, we have that $h_1 = f_1 \mathbin{⨟} g_1$. Since isomorphisms satisfy 2-out-of-3, it follows that vertical morphisms satisfy 2-out-of-3 as well.

Now assume g is cartesian. On directions, $h = f \mathbin{⨟} g$ implies that for every $i \in p(1)$, we have $h_i^\sharp = g_{f_1(i)}^\sharp \mathbin{⨟} f_i^\sharp$. Since $g_{f_1(i)}^\sharp$ is an isomorphism, it follows that every $h_i^\sharp$ is an isomorphism if and only if every $f_i^\sharp$ is an isomorphism, so h is cartesian if and only if f is cartesian. □

Exercise 4.53 (Solution here). Give an example of polynomials p, q, r and maps $p \xrightarrow{f} q \xrightarrow{g} r$ such that f and $f \mathbin{⨟} g$ are cartesian but g is not. ◊

Recall from Exercise 2.53 that for any polynomial p, there is a corresponding function $\pi_p \colon \dot{p}(1) \to p(1)$, i.e. the set of directions mapping to the set of positions. A map of polynomials $(f_1, f^\sharp) \colon p \to q$ can then be described as a function $f_1 \colon p(1) \to q(1)$ along with a function $f^\sharp$ that makes the following diagram in **Set** commute:

$$\begin{tikzcd}
\dot{p}(1) \arrow[d, "\pi_p"'] & \bullet \arrow[l, "f^\sharp"'] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \dot{q}(1) \arrow[d, "\pi_q"] \\
p(1) \arrow[r, equal] & p(1) \arrow[r, "f_1"'] & q(1)
\end{tikzcd} \tag{4.54}$$

Proposition 4.55. Let $f \colon p \to q$ be a morphism of polynomials. The following are equivalent:

1. for each $i \in p(1)$, the induced map $f_i^\sharp$ on directions is a bijection;
2. the diagram in (4.54) can be simplified to a pullback:

$$\begin{tikzcd}
\dot{p}(1) \arrow[r] \arrow[d, "\pi_p"'] \arrow[dr, phantom, "\lrcorner", very near start] & \dot{q}(1) \arrow[d, "\pi_q"] \\
p(1) \arrow[r, "f_1"'] & q(1)
\end{tikzcd}$$

3. Viewed as a natural transformation, f is cartesian, i.e. for any sets A, B and function $g \colon A \to B$, the naturality square

$$\begin{tikzcd}
p(A) \arrow[r, "f_A"] \arrow[d, "p(g)"'] \arrow[dr, phantom, "\lrcorner", very near start] & q(A) \arrow[d, "q(g)"] \\
p(B) \arrow[r, "g_A"'] & q(B)
\end{tikzcd}$$

is a pullback.

Exercise 4.56 (Solution here). Prove Proposition 4.55. ◊

Proposition 4.57. The monoidal structures $+$, $\times$, and $\otimes$ preserve cartesian morphisms.

Proof. Suppose that $f \colon p \to p'$ and $g \colon q \to q'$ 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 $(f \times g)_{i,j}^\sharp$ 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 4.58. Let $f\colon p \to q$ be a morphism and $g\colon q' \to q$ a cartesian morphism. Then the pullback g' of g along p

$$\begin{tikzcd}
p \times_q q' \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & q' \arrow[d, "g"] \\
p \arrow[r, "f"'] & q
\end{tikzcd}$$

is cartesian.

Proof. This follows from Example 4.32, since the pushout of an isomorphism is an isomorphism. □

For any $p \in \mathbf{Poly}$, let $\mathbf{Poly}/p$ denote the slice category, i.e. the category whose objects are maps to p and whose morphisms are commutative triangles.

Definition 4.59. Given a category $\mathcal{C}$ with objects c, d and morphism $f\colon c \to d$ such that all pullbacks along f exist in $\mathcal{C}$, we say that f is *exponentiable* if the functor $f^*\colon \mathcal{C}/d \to \mathcal{C}/c$ given by pulling back along f is a left adjoint.

Theorem 4.60. Cartesian morphisms in **Poly** are exponentiable. That is, if $f\colon p \to q$ is cartesian, then the functor $f^*\colon \mathbf{Poly}/q \to \mathbf{Poly}/p$ given by pulling back along f is a left adjoint:

$$\begin{tikzcd}
\mathbf{Poly}/p \arrow[r, shift right=1.5ex, "f_*"'] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Poly}/q \arrow[l, shift right=1.5ex, "f^*"']
\end{tikzcd}$$

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

$$\begin{tikzcd}
p' \arrow[d, "e"'] & q' \arrow[d, "g"] \\
p \arrow[r, "f"'] & q
\end{tikzcd}$$

We need to define a functor $f_*\colon \mathbf{Poly}/p \to \mathbf{Poly}/q$ 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 give the following formula for f_*p', which we write in larger font for clarity:

$$f_*p' \coloneqq \sum_{j\in q(1)} \sum_{i' \in \prod_{i\in f_1^{-1}(j)} e_1^{-1}(i)} y^{q[j]/\sum_{i\in f_1^{-1}(j)} p'[i'(i)]} \tag{4.61}$$

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

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

□

Example 4.62. Let $p := 2y^2$, $q := y^2 + y^0$, and $f\colon p \to q$ the unique cartesian morphism between them. Then for any $e\colon p' \to p$ over p, (4.61) provides the following description for the pushforward f_*p'.

Over the $j = 2$ position, $f_1^{-1}(2) = 0$ and $q[2] = 0$, so $\prod_{i\in f_1^{-1}(2)} e_1^{-1}(i)$ is an empty product and $q[2]/\sum_{i\in f_1^{-1}(2)} p'[i'(i)]$ is an empty pushout. Hence the corresponding summand of (4.61) is simply $y^0 \cong 1$.

Over the $j = 1$ position, $f_1^{-1}(1) = 2$ and $q[1] = p[1] = p[2] = 2$, so $\prod_{i'\in f_1^{-1}(1)} e_1^{-1}(i) \cong e_1^{-1}(1) \times e_1^{-1}(2)$. For $i' \in e_1^{-1}(1) \times e_1^{-1}(2)$, we have that $q[1]/\sum_{i\in f_1^{-1}(2)} p'[i'(i)] \cong X_{i'}$ in the following pushout square:

$$\begin{tikzcd}
X_{i'} \arrow[dr, phantom, "\lrcorner", very near start] & p'[i'(2)] \arrow[l] \\
p'[i'(1)] \arrow[u] & 2 \arrow[l, "e^\sharp_{i'(1)}"] \arrow[u, "e^\sharp_{i'(2)}"']
\end{tikzcd}$$

Then in sum we have

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

Exercise 4.63 (Solution here). Prove that the unique map $f\colon y \to 1$ is exponentiable.

◊

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

Proof. This follows from (4.33) with $q \coloneqq A$ and $r \coloneqq B$, since pullback of an iso is an iso. □

Theorem 4.65. For any function $f \colon A \to B$, the pullback functor f^* has both a left and a right adjoint

$$\begin{tikzcd} A.\mathbf{Poly} \arrow[r, shift left=4, "f_!"] \arrow[r, phantom, shift left=2, "\Rightarrow"] \arrow[r, phantom, shift right=2, "\Leftarrow"] \arrow[r, shift right=4, "f_*"'] & B.\mathbf{Poly} \arrow[l, "f^*" description] \end{tikzcd} \tag{4.66}$$

Moreover $\otimes$ preserves the op-Cartesian arrows, making this a monoidal ∗-bifibration in the sense of [Shu08, Definition 12.1].

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

$$f_!p \cong \sum_{b \in B} y^{\left(\prod\limits_{a \mapsto b} p[a]\right)} \qquad \text{and} \qquad f_*p \cong \sum_{b \in B} y^{\left(\sum\limits_{a \mapsto b} p[a]\right)} \tag{4.67}$$

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 **??**, 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 $f \colon A \to B$ 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 (4.67) and simultaneously establishes the adjoint triple (4.66).

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 4.57, the monoidal product $\otimes$ preserves cartesian morphisms; thus we will have established the desired monoidal ∗-bifibration in the sense of [Shu08, Definition 12.1] 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 $f \colon p(1) \to B$ and $f' \colon p'(1) \to B'$ are functions then we have

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

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

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

4.5 Summary and further reading

...

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

4.6 Exercise solutions

Solution to Exercise 4.1.

Here $A, B, A', B' \in \mathbf{Set}$.

1. We determine whether various classes of polynomials are closed under addition.
    a) Constant polynomials are closed under addition: given constants A, B, their sum $A + B$ is also a constant polynomial.
    b) Linear polynomials are closed under addition: given linear polynomials Ay, By, their sum $Ay + By \cong (A + B)y$ is also a linear polynomial.
    c) Representable polynomials are *not* closed under addition: for example, y is a representable polynomial, but the sum of y with itself, $2y$, is not.
    d) Monomials are *not* closed under addition: for example, y and $2y^3$ are monomials, but their sum $y + 2y^3$ is not.
2. We determine whether various classes of polynomials are closed under multiplication. The results below follow from Exercise 2.71 #1.
    a) Constant polynomials are closed under multiplication: given constants A, B, their product AB is also a constant polynomial.
    b) Linear polynomials are *not* closed under multiplication: for example, y and $2y$ are linear polynomials, but their product $2y^2$ is not.
    c) Representable polynomials are closed under multiplication: given representables y^A, y^B, their product y^{A+B} is also a representable polynomial.
    d) Monomials are closed under multiplication: given monomials $By^A, B'y^{A'}$, their product $BB'y^{A+A'}$ is also a monomial.
3. We determine whether various classes of polynomials are closed under taking parallel products. The results below follow from Exercise 2.76 #1.
    a) Constant polynomials are closed under taking parallel products: given constants A, B, their parallel product AB is also a constant polynomial.
    b) Linear polynomials are closed under taking parallel products: given linear polynomials Ay, By, their parallel product ABy is also a linear polynomial.
    c) Representable polynomials are closed under taking parallel products: given representables y^A, y^B, their parallel product y^{AB} is also a representable polynomial.
    d) Monomials are closed under taking parallel products: given monomials $By^A, B'y^{A'}$, their parallel product $BB'y^{AA'}$ is also a monomial.

4. We determine whether various classes of polynomials are closed under composition. (Recall that we can think of computing the composite $p \circ q$ of $p, q \in \mathbf{Poly}$ as replacing each appearance of y in p with q.)
   a) Constant polynomials are closed under composition: given constants A, B, their composite $A \circ B \cong A$ is also a constant polynomial.
   b) Linear polynomials are closed under composition: given linear polynomials Ay, By, their composite $Ay \circ By \cong A(By) \cong ABy$ is also a linear polynomial.
   c) Representable polynomials are closed under composition: given representables y^A, y^B, their composite $y^A \circ y^B \cong (y^B)^A \cong y^{BA}$ is also a representable polynomial.
   d) Monomials are closed under taking parallel products: given monomials $By^A, B'y^{A'}$, their composite $By^A \circ B'y^{A'} \cong B(B'y^{A'})^A \cong BB'^A y^{A'A}$ is also a monomial.

Solution to Exercise 4.6.

We complete the proof of Theorem 4.4 by exhibiting three natural isomorphisms, all special cases of (2.42), as follows.

1. By (2.42), we have the natural isomorphism
$$\mathbf{Poly}(A, p) \cong \prod_{a \in A} \sum_{i \in p(1)} 0^{p[i]}.$$
As $0^{p[i]}$ is 1 if $p[i] \cong 0$ and 0 otherwise, it follows that
$$\mathbf{Poly}(A, p) \cong \prod_{a \in A} \{i \in p(1) \mid p[i] \cong 0\} \cong \prod_{a \in A} p(0) \cong \mathbf{Set}(A, p(0)).$$
2. By (2.42), we have the natural isomorphism
$$\begin{aligned}\mathbf{Poly}(p, A) &\cong \prod_{i \in p(1)} \sum_{a \in A} p[i]^0 \\ &\cong \prod_{i \in p(1)} \sum_{a \in A} 1 \\ &\cong \prod_{i \in p(1)} A \\ &\cong \mathbf{Set}(p(1), A).\end{aligned}$$
3. By (2.42), we have the natural isomorphism
$$\begin{aligned}\mathbf{Poly}(Ay, p) &\cong \prod_{a \in A} \sum_{i \in p(1)} 1^{p[i]} \\ &\cong \prod_{a \in A} \sum_{i \in p(1)} 1 \\ &\cong \prod_{a \in A} p(1) \\ &\cong \mathbf{Set}(A, p(1)).\end{aligned}$$

Solution to Exercise 4.7.

Given $p \in \mathbf{Poly}$, we wish to show that $p(1)$ is in bijection with the set of functions $y \to p$. In fact, this follows directly from the Yoneda lemma, but we can also invoke the isomorphism from Exercise 4.6 #3 with $A := 1$ to observe that
$$p(1) \cong \mathbf{Set}(1, p(1)) \cong \mathbf{Poly}(y, p).$$

Solution to Exercise 4.11.

To prove Corollary 4.10, it suffices to exhibit a natural isomorphism
$$\mathbf{Poly}(Ay^B, q) \cong \mathbf{Set}(A, q(B)).$$

Replacing p with y^B in (4.9) from Proposition 4.8, we obtain the natural isomorphism

$$\mathbf{Poly}(Ay^B, q) \cong \mathbf{Set}(A, \mathbf{Poly}(y^B, q)).$$

By the Yoneda lemma, $\mathbf{Poly}(y^B, q)$ is naturally isomorphic to $q(B)$, yielding the desired result.

Solution to Exercise 4.16.

We have the following chain of natural isomorphisms:

$$\begin{aligned}
\Gamma(p \otimes q) &= \mathbf{Poly}(p \otimes q, y) && (3.22)\\
&\cong \mathbf{Poly}(p, [q, y]) && (3.79)\\
&\cong \mathbf{Poly}(p, \Gamma(q) y^{q(1)}) && (3.75)\\
&\cong \mathbf{Set}\big(p(1), \Gamma(q)\big) \times \mathbf{Set}\big(q(1), \Gamma(p)\big). && (4.15)
\end{aligned}$$

Solution to Exercise 4.19.

We are given a monomorphism $f\colon \mathsf{12}y^{\mathsf{12}} \to \mathbb{N}y^{\mathbb{N}}$ from Example 4.18. Let $g\colon \mathbb{N}y^{\mathbb{N}} \to p$ be a dynamical system with yield function $g_1\colon \mathbb{N} \to p(1)$ and update functions $g^\sharp_n\colon p[g_1(n)] \to \mathbb{N}$ for each state $n \in \mathbb{N}$. Then the new composite dynamical system $h \coloneqq f \mathbin{⨟} g$ has a yield function $h_1\colon \mathsf{12} \to p(1)$ which sends each state $i \in \mathsf{12}$ to the output $h_1(i) = g_1(f_1(i)) = g_1(i)$, the same output that the original system yielded in the state $i \in \mathbb{N}$. Meanwhile, the update function for each state $i \in \mathsf{12}$ is a function $h^\sharp_i\colon p[g_1(i)] \to \mathsf{12}$ which, given an input $d \in p[g_1(i)]$, updates the state from i to $h^\sharp_i(d) = f^\sharp_{g_1(i)}(g^\sharp_i(d)) = g^\sharp_i(d) \mod 12$, which is where the original system would have taken the same state to, but reduced modulo 12. In other words, the new system behaves like the old system but with only the states in $\mathsf{12} \subseteq \mathbb{N}$ retained, and on any input that would have caused the old system to move to a state outside of $\mathsf{12}$, the new system moves to the equivalent state (modulo 12) within $\mathsf{12}$ instead.

Solution to Exercise 4.21.

Given $p \in \mathbf{Poly}$ and a map $f\colon p \to y$, we will use Proposition 4.20 to show that either f is an epimorphism or $p = 0$. First, note that $f_1\colon p(1) \to 1$ must be an epimorphism unless $p(1) \cong 0$, in which case $p = 0$. Next, note that the induced function

$$f^\flat\colon 1 \to \prod_{i \in p(1)} p[i]$$

from (2.50) must be a monomorphism. So it follows from Proposition 4.20 that either f is an epimorphism or $p = 0$.

Solution to Exercise 4.22.

Given sets A and B, by Proposition 4.20, a morphism $f\colon y^A + y^B \to y^{AB}$ is an epimorphism if its on-positions function $f_1\colon 2 \to 1$ is an epimorphism (which must be true) and if the induced function

$$f^\flat\colon AB \to \prod_{i \in 2} (y^A + y^B)[i] \cong AB$$

is a monomorphism. If we take the on-directions functions $AB \to A$ and $AB \to B$ of f to be the canonical projections, then the induced function $f^\flat\colon AB \to AB$ would be the identity, which is indeed a monomorphism. So f would be an epimorphism.

Solution to Exercise 4.23.

Let $f\colon p \to q$ be a morphism in **Poly** that is both a monomorphism and an epimorphism. We claim that f is an isomorphism. By Proposition 4.17 and Proposition 4.20, the on-positions function $f_1\colon p(1) \to q(1)$

is both a monomorphism and an epimorphism, so it is an isomorphism. Meanwhile, Proposition 4.20 says that, for each $j \in q(1)$, the induced function

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

is a monomorphism. As f_1 is an isomorphism, it follows that for each $i \in p(1)$, the function

$$f_{f_1(i)}^\flat \colon q[f_1(i)] \to p[i]$$

is a monomorphism. But this is just the on-directions function $f_i^\sharp$ of f. From Proposition 4.17, we also know that $f_i^\sharp$ is an epimorphism. It follows that every on-directions function of f is an isomorphism. Hence f itself is an isomorphism.

Solution to Exercise 4.24.

**

Solution to Exercise 4.27.

We use (4.25) to compute various exponentials. Here $p \in \mathbf{Poly}$ and $A, B \in \mathbf{Set}$.

1. We have that p^0 is an empty product, so $p^0 \cong 1$ as expected.
2. We have that $p^1 \cong p \circ (y + 0) \cong p$, as expected.
3. We have that $1^p \cong \prod_{i \in p(1)} 1 \circ (y + p[i]) \cong 1$, as expected.
4. We have that $A^p \cong \prod_{i \in p(1)} A \circ (y + p[i]) \cong A^{p(1)}$.
5. We have that $y^y \cong y \circ (y + 1) \cong y + 1$.
6. We have that $y^{4y} \cong \prod_{j \in 4} y \circ (y + 1) \cong (y + 1)^4 \cong y^4 + 4y^3 + 6y^2 + 4y + 1$.
7. We have that $(y^A)^{y^B} \cong (y^A) \circ (y + B) \cong (y + B)^A \cong \sum_{f \colon A \to 2} B^{f^{-1}(1)} y^{f^{-1}(2)}$.

Solution to Exercise 4.29.

By Theorem 4.28, there is a natural isomorphism

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

Under this isomorphism, there exists a map $\operatorname{eval} \colon p^q \times q \to p$ corresponding to the identity map on p^q. The map eval is the canonical evaluation map.

Solution to Exercise 4.31.

1. The morphism $e \colon p' \to p$ can be characterized as follows. The on-positions function $e_1 \colon p'(1) \to p(1)$ is the equalizer of $f_1, g_1 \colon p(1) \rightrightarrows q(1)$ in $\mathbf{Set}$. In particular, e_1 is the canonical inclusion that sends each element of $p'(1)$ to the same element in $p(1)$. Then for each $i \in p'(1)$, the on-directions function $e_i^\sharp \colon p[i] \to p'[i]$ is the coequalizer of $f_i^\sharp, g_i^\sharp \colon q[f_1(i)] \rightrightarrows p[i]$ in $\mathbf{Set}$.
2. To show that $e \fatsemi f = e \fatsemi g$, it suffices to show that both sides are equal on positions and on directions. On positions, e_1 is defined to be the equalizer of f_1 and g_1, so $e_1 \fatsemi f_1 = e_1 \fatsemi g_1$. Then for each $i \in p'(1)$, the on-directions function $e_i^\sharp$ is defined to be the coequalizer of $f_i^\sharp$ and $g_i^\sharp$, so $f_i^\sharp \fatsemi e_i^\sharp = g_i^\sharp \fatsemi e_i^\sharp$.
3. To show that e is the equalizer of f and g, it suffices to show that for any $r \in \mathbf{Poly}$ and map $a \colon r \to p$ satisfying $a \fatsemi f = a \fatsemi g$, there exists a unique map $h \colon r \to p'$ for which $a = h \fatsemi e$, so that the following diagram commutes.

$$\begin{tikzcd}
p' \arrow[r, "e"] & p \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & q \\
r \arrow[u, dashed, "h"] \arrow[ur, "a"']
\end{tikzcd}$$

In order for $a = h \mathbin{⨟} e$ to hold, we must have $a_1 = h_1 \mathbin{⨟} e_1$ on positions. But we have that $a_1 \mathbin{⨟} f_1 = a_1 \mathbin{⨟} g_1$, so by the universal property of $p'(1)$ and the map e_1 as the equalizer of f_1 and g_1 in **Set**, there exists a unique h_1 for which $a_1 = h_1 \mathbin{⨟} e_1$. Hence h is uniquely characterized on positions. In particular, it must send each $k \in r(1)$ to $a_1(k) \in p'(1)$.

Then for $a = h \mathbin{⨟} e$ to hold on directions, we must have that $a^\sharp_k = e^\sharp_{a_1(k)} \mathbin{⨟} h^\sharp_k$ for each $k \in r(1)$. But we have that $f^\sharp_{a_1(k)} \mathbin{⨟} a^\sharp_{a_1(k)} = g^\sharp_{a_1(k)} \mathbin{⨟} a^\sharp_{a_1(k)}$, so by the universal property of $p'[a_1(k)]$ and the map $e^\sharp_{a_1(k)}$ as the coequalizer of $f^\sharp_{a_1(k)}$ and $g^\sharp_{a_1(k)}$ in **Set**, there exists a unique $h^\sharp_k$ for which $a^\sharp_k = e^\sharp_{a_1(k)} \mathbin{⨟} h^\sharp_k$, so that the diagram below commutes.

$$\begin{tikzcd}
p'[a_1(k)] \arrow[d, dashed, "h^\sharp_k"'] & p[a_1(k)] \arrow[l, "e^\sharp_{a_1(k)}"'] \arrow[dl, "a^\sharp_k"] & q[f_1(a_1(k))] \arrow[l, shift right, "f^\sharp_{a_1(k)}"'] \arrow[l, shift left, "g^\sharp_{a_1(k)}"] \\
r[k] & &
\end{tikzcd}$$

Hence h is also uniquely characterized on directions, so it is unique overall. Moreover, we have shown that we can define h on positions so that $a_1 = h_1 \mathbin{⨟} e_1$, and that we can define h on directions such that $a^\sharp_k = e^\sharp_{a_1(k)} \mathbin{⨟} h^\sharp_k$ for all $k \in r(1)$. It follows that there exists h for which $a = h \mathbin{⨟} e$.

Solution to Exercise 4.34.

Here $p \in \mathbf{Poly}$.

1. The canonical morphism $\eta\colon p \to p(1)$ is the identity $\eta_1\colon p(1) \to p(1)$ on positions and the empty function on directions.
2. On positions, we have that $p_i(1)$ along with f_1 and g_1 form the following pullback square in **Set**:

$$\begin{tikzcd}
p_i(1) \arrow[r, "g_1"] \arrow[d, "f_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & p(1) \arrow[d, equal] \\
1 \arrow[r, "i"'] & p(1)
\end{tikzcd}$$

So $p_i(1) := \{(a, i') \in 1 \times p(1) \mid i = i'\} = \{(1, i)\}$, with f_1 uniquely determined and g_1 picking out $i \in p(1)$. Then on directions, we have that $p_i[(1, i)]$ along with $f^\sharp_{(1,i)}$ and $g^\sharp_{(1,i)}$ form the following pushout square in **Set**:

$$\begin{tikzcd}
p_i[(1,i)] \arrow[dr, phantom, "\lrcorner", very near start] & p[i] \arrow[l, "g^\sharp_{(1,i)}"'] \\
0 \arrow[u, "f^\sharp_{(1,i)}"] & 0 \arrow[l, "!"] \arrow[u, "!"']
\end{tikzcd}$$

So $p_i[(1, i)] := p[i]$, with $f^\sharp_{(1,i)}$ uniquely determined and $g^\sharp_{(1,i)}$ as the identity. It follows that $p_i := \{(1, i)\}y^{p[i]} \cong y^{p[i]}$, where $f\colon p_i \to 1$ is uniquely determined and $g\colon p_i \to p$ picks out $i \in p(1)$ on positions and is the identity on $p[i]$ on directions.

Solution to Exercise 4.35.

1. There are many possible answers, but one morphism $f\colon q \to r$, on positions, sends $1 \in q(1)$ (corresponding to y^2) to $2 \in r(1)$ (corresponding to 1) and $2 \in q(1)$ (corresponding to y) to $1 \in r(1)$ (corresponding to y). Then the on-directions functions $f^\sharp_1\colon 0 \to 2$ and $f^\sharp_2\colon 1 \to 1$ are uniquely determined. Another morphism $f'\colon q' \to r$, on positions, sends $1 \in q'(1)$ (corresponding to one of the y^3 terms) to $2 \in r(1)$ and both $2 \in q'(1)$ (corresponding to the other y^3 term) and $3 \in q'(1)$ (corresponding to the y^2 term) to $1 \in r(1)$. Then the on-directions function $(f')^\sharp_1\colon 0 \to 3$ is uniquely determined, while we can let $(f')^\sharp_2\colon 1 \to 3$ pick out 3 and $(f')^\sharp_3\colon 1 \to 2$ pick out 1.

2. We compute the pullback p along with the morphisms $g\colon p \to q$ and $g'\colon p \to q'$ of $q \xrightarrow{f} r \xleftarrow{f'} q'$ by following Example 4.32. We can compute $p(1)$ by taking the pullback in **Set**:
$$p(1) \coloneqq \{(i,i') \in \mathsf{2} \times \mathsf{3} \mid f_1(i) = f_1'(i)\} = \{(1,1),(2,2),(2,3)\}.$$
Moreover, the on-positions functions g_1 and g_1' send each pair in $p(1)$ to its left component and its right component, respectively.
To compute the set of directions at each position of p, we must compute a pushout. At $(1,1)$, we have $r[f_1(1)] = r[f_1'(1)] = r[2] = \mathsf{0}$, so the pushout $p[(1,1)]$ is just the sum $q[1] + q'[1] = \mathsf{2} + \mathsf{3} \cong \mathsf{5}$. Moreover, the on-directions functions $g^\sharp_{(1,1)}$ and $(g')^\sharp_{(1,1)}$ are the canonical inclusions $\mathsf{2} \to \mathsf{2} + \mathsf{3}$ and $\mathsf{3} \to \mathsf{2} + \mathsf{3}$.
At $(2,2)$, we have $r[f_1(2)] = r[f_1'(2)] = r[1] = \mathsf{1}$, with $f_2^\sharp$ picking out $1 \in \mathsf{1} = q[2]$ and $(f')^\sharp_2$ picking out $3 \in \mathsf{3} = q'[2]$. So the pushout $p[(2,2)]$ is the set $\mathsf{1} + \mathsf{3} = \{(1,1),(2,1),(2,2),(2,3)\}$ but with $(1,1)$ identified with $(2,3)$; we can think of it as the set of equivalence classes $p[(2,2)] \cong \{\{(1,1),(2,3)\},\{(2,1)\},\{(2,2)\}\} \cong \mathsf{3}$. Moreover, the on-directions function $g^\sharp_{(2,2)}$ maps $1 \mapsto \{(1,1),(2,3)\}$, while the on-directions function $(g')^\sharp_{(2,2)}$ maps $1 \mapsto \{(2,1)\}, 2 \mapsto \{(2,2)\}$, and $3 \mapsto \{(1,1),(2,3)\}$.
Finally, at $(2,3)$, we have $r[f_1(2)] = r[f_1'(3)] = r[1] = \mathsf{1}$, with $f_2^\sharp$ still picking out $1 \in \mathsf{1} = q[2]$ and $(f')^\sharp_3$ picking out $1 \in \mathsf{2} = q'[3]$. So the pushout $p[(2,3)]$ is the set $\mathsf{1} + \mathsf{2} = \{(1,1),(2,1),(2,2)\}$ but with $(1,1)$ identified with $(2,1)$; we can think of it as the set of equivalence classes $p[(2,3)] \cong \{\{(1,1),(2,1)\},\{(2,2)\}\} \cong \mathsf{2}$. Moreover, the on-directions function $g^\sharp_{(2,3)}$ maps $1 \mapsto \{(1,1),(2,1)\}$, while the on-directions function $(g')^\sharp_{(2,3)}$ maps $1 \mapsto \{(1,1),(2,1)\}$ and $2 \mapsto \{(2,2)\}$.
It follows that $p \cong y^5 + y^3 + y^2$, with g and g' as described.

Solution to Exercise 4.37.

1. We define a map $g\colon q \to q'$ as follows. The on-positions function $g_1\colon q(1) \to q'(1)$ is the coequalizer of $s_1, t_1\colon p(1) \rightrightarrows q(1)$. In particular, g_1 sends each vertex in $q(1)$ to its corresponding connected component in $q'(1) = C$. Then for each $v \in q(1)$, if we let its corresponding connected component be $c \coloneqq g_1(v)$, we can define the on-directions function $g^\sharp_v\colon q'[c] \to q[v]$ to be the projection from the limit $q'[c]$ to its component $q[v]$.
2. To show that $s \fatsemi g = t \fatsemi g$, we must show that both sides are equal on positions and on directions. The on-positions function g_1 is defined to be the coequalizer of s_1 and t_1, so $s_1 \fatsemi g_1 = t_1 \fatsemi g_1$. So it suffices to show that for all $e \in p(1)$, if we let its corresponding connected component be $c \coloneqq g_1(s_1(e)) = g_1(t_1(e))$, then the following diagram of on-directions functions commutes:
$$\begin{tikzcd}
& q[s_1(e)] \arrow[dl, "s_e^\sharp"'] & \\
p[e] & & q'[c] \arrow[ul, "g^\sharp_{s_1(e)}"'] \arrow[dl, "g^\sharp_{t_1(e)}"] \\
& q[t_1(e)] \arrow[ul, "t_e^\sharp"] &
\end{tikzcd}$$
But this is automatically true by the definition of $q'[c]$ as a limit—specifically the limit of a functor with $s_e^\sharp$ and $t_e^\sharp$ in its image—and the definitions of $g^\sharp_{s_1(e)}$ and $g^\sharp_{t_1(e)}$ as projections from this limit.
3. To show that g is the coequalizer of s and t, it suffices to show that for any $r \in$ **Poly** and map $f\colon q \to r$ satisfying $s \fatsemi f = t \fatsemi f$, there exists a unique map $h\colon q' \to r$ for which $f = g \fatsemi h$, so that the following diagram commutes.
$$\begin{tikzcd}
p \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] & q \arrow[r, "g"] \arrow[dr, "f"'] & q' \arrow[d, dashed, "h"] \\
& & r
\end{tikzcd}$$

In order for $f = g \fatsemi h$ to hold, we must have $f_1 = g_1 \fatsemi h_1$ on positions. But we have that $s_1 \fatsemi f_1 = t_1 \fatsemi f_1$, so by the universal property of $q'(1)$ and the map g_1 as the coequalizer of s_1 and t_1 in **Set**, there exists a unique h_1 for which $f_1 = g_1 \fatsemi h_1$. Hence h is uniquely characterized on positions. In particular, it must send each connected component $c \in q'(1)$ to the element in $r(1)$ to which f_1 sends every vertex $v \in V_c = g_1^{-1}(c)$ that lies in the connected component c.

Then for $f = g \fatsemi h$ to hold on directions, we must have that $f_v^\sharp = h^\sharp_{g_1(v)} \fatsemi g_v^\sharp$ for each $v \in q(1)$. Put another way, given $c \in q'(1)$, we must have that $f_v^\sharp = h_c^\sharp \fatsemi g_v^\sharp$ for every $v \in V_c$. But $s \fatsemi f = t \fatsemi f$ implies that for each $e \in E_c = s_1^{-1}(g_1^{-1}(c)) = t_1^{-1}(g_1^{-1}(c)) \subseteq p(1)$, the following diagram of on-directions functions commutes:

$$\begin{tikzcd}
 & q[s_1(e)] \arrow[dl, "s_e^\sharp"'] & \\
p[e] & & r[f_1(v)] \arrow[ul, "f^\sharp_{s_1(e)}"'] \arrow[dl, "f^\sharp_{t_1(e)}"] \\
 & q[t_1(e)] \arrow[ul, "t_e^\sharp"] &
\end{tikzcd}$$

It follows that $r[f_1(v)]$ together with the maps $\{f_v^\sharp\}_{v \in V_c}$ form a cone over the functor F. So by the universal property of the limit $q'[c]$ of F with projection maps $\{g_v^\sharp\}_{v \in V_c}$, there exists a unique $h_c^\sharp \colon r[f_1(v)] \to q'[c]$ for which $f_v^\sharp = h_c^\sharp \fatsemi g_v^\sharp$ for every $v \in V_c$. Hence h is also uniquely characterized on directions, so it is unique overall. Moreover, we have shown that we can define h on positions so that $f_1 = g_1 \fatsemi h_1$, and that we can define h on directions such that $f_v^\sharp = h_c^\sharp \fatsemi g_v^\sharp$ for all $c \in q'(1)$ and $v \in V_c$. It follows that there exists h for which $f = g \fatsemi h$.

Solution to Exercise 4.40.

1. We characterize the map $\epsilon \colon p(1)y \to p$ as follows. On positions, it is the identity on $p(1)$. Then for each $i \in p(1)$, on directions, it is the unique map $p[i] \to 1$.
2. We characterize the map $\eta \colon p \to y^{\Gamma(p)}$ as follows. On positions, it is the unique map $p(1) \to 1$. Then for each $i \in p(1)$, on directions, it is the canonical projection $\Gamma(p) \cong \prod_{i' \in p(1)} p[i'] \to p[i]$.
3. Showing that (4.41) is a pushout square is equivalent to showing that, in the diagram

$$\begin{tikzcd}
 & y \arrow[d, "\iota"] \arrow[dr, "!"] & \\
p(1)y \arrow[ur, "!"] \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] \arrow[dr, "\epsilon"'] & y + p \arrow[r, "g"] & y^{\Gamma(p)} \\
 & p \arrow[u, "\iota'"] \arrow[ur, "\eta"'] &
\end{tikzcd} \tag{4.68}$$

   in which ι, ι' are the canonical inclusions and the four triangles commute, $y^{\Gamma(p)}$ equipped with the morphism g is the coequalizer of s and t. To do so, we apply Theorem 4.36 to compute the coequalizer q' of s and t. The set of positions of q' is the coequalizer of $s_1 = (! \fatsemi \iota)_1$, which sends every $i \in p(1)$ to the position of $y + p$ corresponding to the summand y, and $t_1 = (\epsilon \fatsemi \iota')_1$, which sends each $i \in p(1)$ to the corresponding position in the summand p of $y + p$. It follows that the coequalizer of s_1 and t_1 is 1, so $q'(1) \cong 1$.

   Then the set of directions of q' at its sole position is the limit of the functor F whose image consists of morphisms of the form $1 \to 1$ or $p[i] \to 1$ for every $i \in p(1)$. It follows that the limit of F is just a product, namely $\prod_{i \in p(1)} p[i] \cong \Gamma(p)$. Hence $q' \cong y^{\Gamma(p)}$, as desired.

   It remains to check that the upper right and lower right triangles in (4.68) commute. The upper right triangle must commute by the uniqueness of morphisms $y \to y^{\Gamma(p)}$; and the lower right triangle must commute on positions. Moreover, the on-directions function of the coequalizer morphism g at each position $i \in p(1) \subseteq (y + p)(1)$ must be the canonical projection $\Gamma(p) \to p[i]$, which matches the behavior of the corresponding on-directions function of η; hence the lower right triangle also commutes on directions.

Solution to Exercise 4.53.

Consider the maps $y \xrightarrow{f} y^2 + y \xrightarrow{g} y$ where f is the canonical inclusion and g is uniquely determined on positions and picks out $1 \in 2$ and $1 \in 1$ on directions. Then the only on-directions function of f is a function $1 \to 1$, an isomorphism, so f is cartesian. Meanwhile, one of the on-directions functions of g is a function $1 \to 2$, which is not an isomorphism, so g is not cartesian. Finally, $f \mathbin{⨟} g$ can only be the unique morphism $y \to y$, namely the identity, which is cartesian.

Solution to Exercise 4.56.

**

Solution to Exercise 4.63.

Choose $p \in \mathbf{Poly}$ and $q' \in \mathbf{Poly}/y$. Then there is $q \in \mathbf{Poly}$ such that $q' \cong qy$, equipped with the projection $qy \to y$. The pushforward is given by the exponential

$$f_*(qy) \coloneqq q^y$$

from the cartesian closure; see (4.25). Indeed, we have

$$\begin{aligned}\mathbf{Poly}/y(f^*p, qy) &\cong \mathbf{Poly}/y(py, qy)\\ &\cong \mathbf{Poly}(py, q)\\ &\cong \mathbf{Poly}(p, q^y).\end{aligned}$$

Part II

A different category of categories

Chapter 5

The composition product

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 interfaces and wiring patterns based on their internal states.

In this part 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 Part III 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 part is to first introduce what is perhaps the most interesting monoidal structure on **Poly**, namely the composition product; we do so in Chapter 5. 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 Chapter 6 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 Chapter 7 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.

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

But we touched upon one thing—what in some sense is the most interesting part of the story—only briefly. 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?

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 5.2—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 **??**. 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 (2.35), we'll continue to denote polynomials with the following notation

$$p \cong \sum_{i \in p(1)} y^{p[i]}, \tag{5.1}$$

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 directions at position i.

5.1 Defining the composition product

We begin with the definition of composition product.

Proposition 5.2. 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 isomorphism:

$$p \circ q \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in q(1)} \prod_{e \in q[j]} y. \tag{5.3}$$

Proof. We can rewrite (5.1) 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 (5.3) is indeed the formula for their composite. To see this is a polynomial, we use (2.24), which says we can rewrite the $\prod \sum$ in (5.3) as a $\sum \prod$ to obtain

$$p \circ q \cong \sum_{i \in p(1)} \; \sum_{j_i \colon p[i] \to q(1)} y^{\sum_{d \in p[i]} q[j_i(d)]} \tag{5.4}$$

(written slightly bigger for clarity), which 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

$$p \mathbin{\triangleleft} q \coloneqq p \circ q.$$

The symbol $\triangleleft$ looks a bit like the composition symbol, in that it is an open shape, and when writing quickly by hand, it's okay if it morphs into a $\circ$. But $\triangleleft$ highlights the asymmetry of composition, in contrast with the other monoidal structures on **Poly** we've encountered, and 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 5.2 in the new notation:

$$p \mathbin{\triangleleft} q \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in q(1)} \prod_{e \in q[j]} y. \tag{5.5}$$

$$\begin{array}{c} \begin{array}{|c|}\hline e : q[j] \\ \hline j : q(1) \\ \hline \end{array} \\[1em] \begin{array}{|c|}\hline d : p[i] \\ \hline i : p(1) \\ \hline \end{array} \end{array} \quad \cong \quad \begin{array}{|c|}\hline (d : p[i], e : q[j(d)]) \\ \hline (i : p(1), j : p[i] \to q(1)) \\ \hline \end{array}$$

Exercise 5.6 (Solution here). Let's consider (5.4) piece by piece, with concrete polynomials $p \coloneqq y^2 + y^1$ and $q \coloneqq y^3 + 1$.

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

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

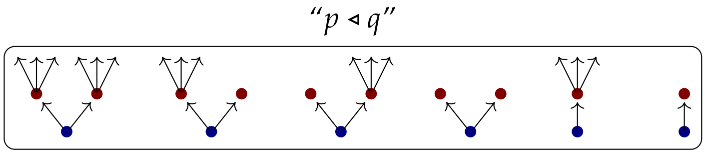
Exercise 5.7 (Solution here).

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, the forest of the composition product $p \triangleleft q$ is given by gluing corollas from the forest of q onto the leaves of corollas from the forest of p in every possible way. Let's say $p := y^2 + y$ and $q := y^3 + 1$, whose corolla forests we draw as follows:

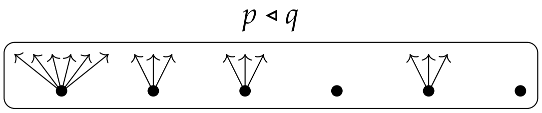
(5.8)

Then their composite $p \triangleleft q$ is obtained by stacking the corollas like so:



(5.9)

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 $y^6 + 3y^3 + 2$. We put the $p \triangleleft q$ in scare quotes because, to be pedantic, we should smash the two levels together and redraw $p \triangleleft q$ as follows:



(5.10)

Usually, we will prefer the (5.9) style rather than the more pedantic (5.10) style.

Exercise 5.11 (Solution here). Use p, q as in (5.8) 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 $r \triangleleft r \triangleleft r$. ◊

Proposition 5.12. For any polynomials $p, q \in \mathbf{Poly}$, there is a cartesian map

$$p \otimes q \to p \triangleleft q.$$

It constitutes a lax monoidal functor $(\mathbf{Poly}, y, \otimes) \to (\mathbf{Poly}, y, \triangleleft)$.

Proof. For p, q, the map is given by

$$(i \in p(1), j \in q(1)) \mapsto (i, d \in p[i] \mapsto j, (d, e) \in p[i] \times q[j] \mapsto (d, e))$$

□

Exercise 5.13 (Solution here). 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. $p \otimes y^A \cong p \triangleleft y^A$? ◊

Here is a curious fact relating composition and the closure operation adjoint to $\otimes$. Recall that for any two polynomials p, q, there is a polynomial $[p, q] \in \mathbf{Poly}$.

Proposition 5.14. For any set $A \in \mathbf{Set}$ and any polynomial $p \in \mathbf{Poly}$, there is an isomorphism

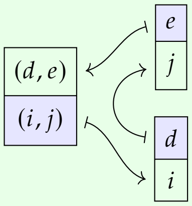
$$y^A \triangleleft p \cong [Ay, p].$$

Proof.

$$\begin{aligned} y^A \triangleleft p &\cong \prod_{a \in A} \sum_{i \in p(1)} y^{p[i]} \\ &\cong \sum_{i \colon A \to p(1)} \prod_{a \in A} y^{p[i(a)]} \\ &\cong \sum_{i \colon A \to p(1)} y^{\sum_{a \in A} p[i(a)]} \\ &\cong \sum_{\varphi \colon Ay \to p} y^{\sum_{a \in Ay(1)} p[\varphi_1(a)]} \\ &\cong [Ay, p] \end{aligned}$$

□

Example 5.15. For any p and q there is an interesting map $o_{p,q}\colon p\otimes q \to p \triangleleft q$ that orders the operation. It looks like this:



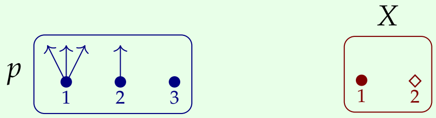
In other words, $p \triangleleft q$ is allowed to have j depend on d, whereas $p \otimes q$ is not; the map is in some sense the inclusion of the order-independent part. And of course we can flip the order using the symmetry $q \otimes p \cong p \otimes q$. This is, we just as well have a map $p \otimes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

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.

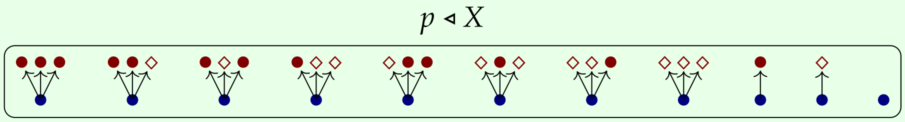
This can be used in the following way. Maps $p \to q \triangleleft r$ into composites are fairly easy to diagram and understand, whereas maps $q \triangleleft r \to p$ are not so easy to think about. However, given such a map, one may always compose it with $o_{q,r}$ to obtain a map $q \otimes r \to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

Example 5.16. For any set X and polynomial p, we can take $p(X) \in \mathbf{Set}$; indeed $p\colon \mathbf{Set} \to \mathbf{Set}$ 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 $X := 2$.



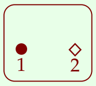
Let's see how $(y^3 + y + 1) \triangleleft 2$ looks.



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.

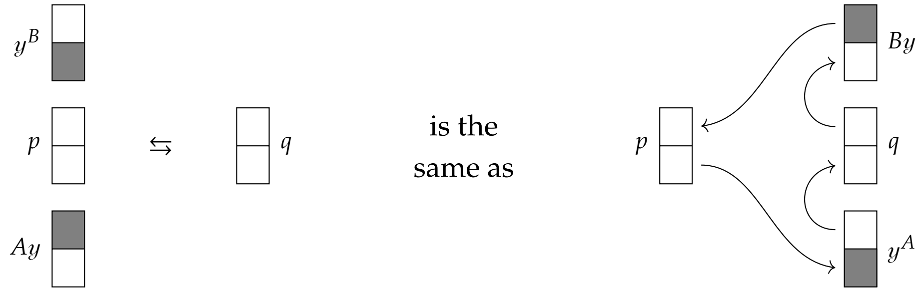
Exercise 5.17 (Solution here).

1. Choose a polynomial p and draw $p \triangleleft 1$ in the style of Example 5.16.
2. Show that if X is a set (considered as a constant polynomial) and p is any polynomial, then $X \triangleleft p \cong X$.
3. Show 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 5.18. 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

$$\mathbf{Poly}\left(Ay \triangleleft p, q\right) \cong \mathbf{Poly}\left(p, y^A \triangleleft q\right) \tag{5.19}$$

and that

$$\mathbf{Poly}\left(p \triangleleft y^B, q\right) \cong \mathbf{Poly}\left(p, q \triangleleft By\right) \tag{5.20}$$

For Eq. (5.19), 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. (5.20), 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 5.21 (Solution here). 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:

$$\mathbf{Poly}(Ay^B, p) \cong^? \mathbf{Poly}(A, p \triangleleft B) \tag{5.22}$$

If so, why? If not, give a counterexample. ◊

Exercise 5.23 (Solution here). 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 $p \triangleleft y$ 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 $y \triangleleft p \cong p \cong p \triangleleft y$. ◊

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

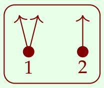
The technical claim is that $\triangleleft$ is a monoidal product, which means that it's well-behaved; in particular, it's functorial, associative, and unital. In fact, all of this comes from general theory: for any category $\mathcal{C}$, the category whose objects are functors $\mathcal{C} \to \mathcal{C}$ and whose morphisms are natural transformations is a monoidal category, with functor composition as the monoidal product. For us, $\mathcal{C} := \mathbf{Set}$; yet we are only using polynomial functors, not all functors, so there is a tiny bit more to check, but it's accomplished by Proposition 5.2 and the fact that the identity functor $\mathbf{Set} \to \mathbf{Set}$ is a polynomial (it's y).

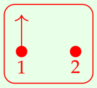
However, even though the formal theory of functors and natural transformations knocks the monoidality of $\triangleleft$ out of the park, it is still useful to discuss how it acts on morphisms in terms of positions and directions.

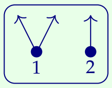
For any $f \colon p \to p'$ and $g \colon q \to q'$, we should obtain a morphism $(f \triangleleft g) \colon (p \triangleleft q) \to (p' \triangleleft q')$. This is actually quite an impressive operation! It threads back and forth in a fascinating way.

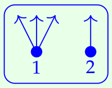
Recall from Example 2.44 that we can think of $f = (f_1, f^\sharp)$ as a way to delegate decisions from p to p'. Every decision (i.e. position) $i \in p(1)$ is assigned a decision $f_1(i) \in p'(1)$. Then every option $d \in p'[f_1(i)]$ there is passed back to an option $f^\sharp(d) \in p[i]$. Let's start with an example and then give the general method.

Example 5.24. Let's take $p \coloneqq y^2 + y$, $q \coloneqq y^2 + y$, $p' \coloneqq y^3 + y$, and $q' \coloneqq y + 1$.

$q =$ 

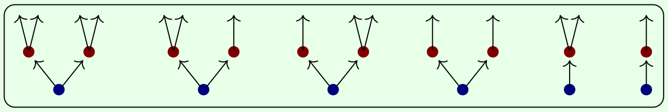
$q' =$ 

$p =$ 

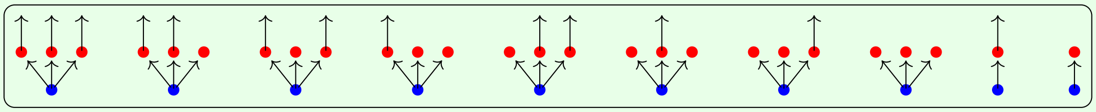
$p' =$ 

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 $p' \triangleleft q'$.

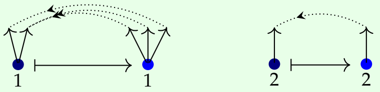
$p \triangleleft q$

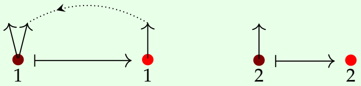


$p' \triangleleft q'$



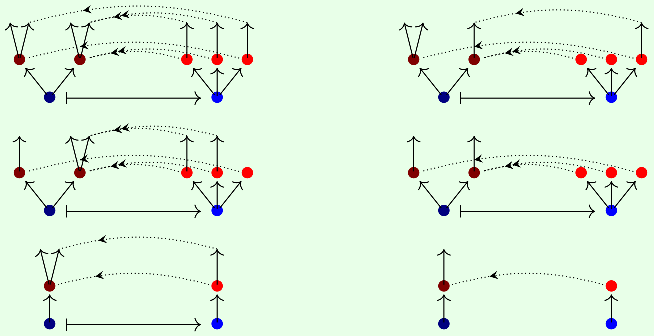
Okay, now suppose someone gives us delegations (i.e. morphisms) $f\colon p \to p'$ and $g\colon q \to q'$. Let's just pick something relatively at random:

$f\colon p \to p'$ 

$g\colon q \to q'$ 

Then we can form the induced delegation $f \triangleleft g\colon (p \triangleleft q) \to (p' \triangleleft q')$ 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 5.25 (Solution here). With p, q, p', q' and f, g as in Example 5.24, draw $g \triangleleft f \colon (q \triangleleft p) \to (q' \triangleleft p')$ in terms of trees as in the example. ◊

Exercise 5.26 (Solution here). Suppose p, q, and r are polynomials and you're given arbitrary morphisms $f \colon q \to p \triangleleft q$ and $g \colon q \to q \triangleleft r$. Does the following diagram necessarily commute?

$$\begin{tikzcd}
q \arrow[r, "g"] \arrow[d, "f"'] \arrow[dr, phantom, "?"] & q \triangleleft r \arrow[d, "f \triangleleft r"] \\
p \triangleleft q \arrow[r, "p \triangleleft g"'] & p \triangleleft q \triangleleft r
\end{tikzcd}$$

That is, do we have $(p \triangleleft g) \circ f \overset{?}{=} (f \triangleleft r) \circ g$? ◊

5.2.1 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 5.27 (Solution here).

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. But the easiest way to think of $p^{\triangleleft n}$ is as height-n trees with node labels and branching as dictated by p.

Exercise 5.28 (Solution here). Let $p, q \in$ **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." ◊

5.3 Working with composites

We need a way of talking about maps to composites.[1] 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

$$(f_1, f_2, \ldots, f_k, f^\sharp) \colon p \longrightarrow q_1 \triangleleft q_2 \triangleleft \cdots \triangleleft q_k, \tag{5.29}$$

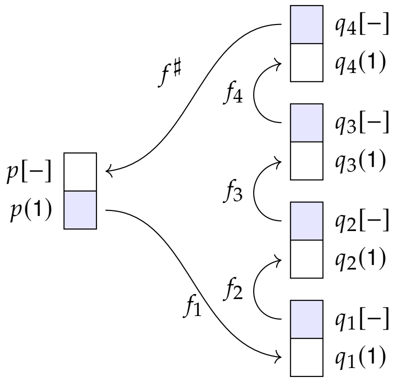
---

[1] It would be nice to also have a nice way of talking about maps *out of* composites; however that is more difficult.

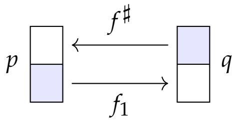
where

$$
\begin{aligned}
&f_1 : p(1) \to q_1(1),\\
&f_2 : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to q_2(1),\\
&f_3 : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to (e_2 \in q_2[f_2(i, e_1)]) \to q_3(1),\\
&f_k : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to \cdots \to (e_{k-1} \in q_{k-1}[f_{k-1}(i, e_1, \ldots, e_{k-2})]) \to q_k(1),\\
&f^\sharp : (i \in p(1)) \to (e_1 \in q_1[f_1(i)]) \to \cdots \to (e_k \in q_k[f_k(i, e_1, \ldots, e_{k-1})]) \to p[i]
\end{aligned}
\tag{5.30}
$$

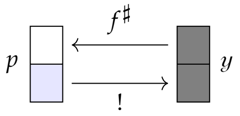
Here's a picture for the $k = 4$ case:



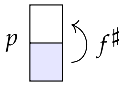
A few considerations might make (5.30) less scary. First of all, we're usually interested in the cases $k = 0, 1, 2$. The case $k = 1$ is just our $(f_1, f^\sharp)$ notation



The case $k = 0$ says that a map $p \to y$ is just a function $f^\sharp \in \prod_{i \in p(1)} p[i]$

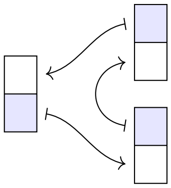


which we can rewrite simply as



(5.31)

The case $k = 2$ looks like this



(5.32)

and is considered in Example 5.33. But before we get there, let's think about (5.30) 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 $f \colon p \to q_1 \triangleleft \cdots \triangleleft q_k$, 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 $f^\sharp(i, e_1, \ldots, e_k) \in p[i]$.

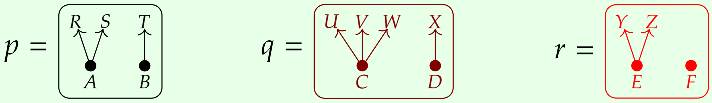
A morphism $p \to q_1 \triangleleft \cdots \triangleleft q_k$ is a multi-step policy for p to make decisions by asking for decisions from q_1 then q_2, etc., all the way until q_k, and interpreting the results.

Example 5.33 (Maps $p \to q \triangleleft r$). By (5.30), a morphism $p \to q \triangleleft r$ can be specified by a tuple $(f_1, f_2, f^\sharp)$, where $f_1 \colon p(1) \to q(1)$ and $f_2, f^\sharp$ are a little more involved because they are dependent functions.

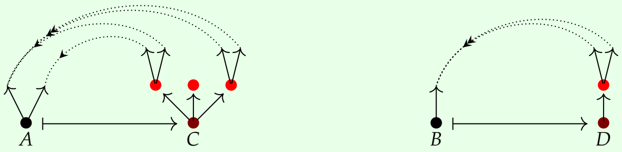
The dependent function f_2 takes as input a pair (i, d) where $i \in p(1)$ and $d \in q[f_1(i)]$, and it outputs an element of $r(1)$.

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 $r := \{E\}y^{\{Y,Z\}} + \{F\}$.



Here is a picture of a map $p \to q \triangleleft r$:

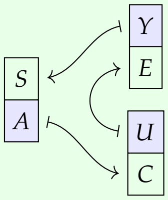


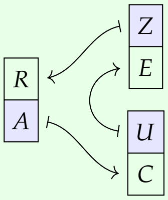
If we write it as $(f_1, f_2, f^\sharp) \colon p \to q \triangleleft r$ then we have

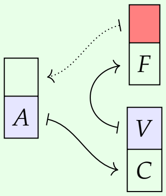
$$f_1(A) = C, \quad f_1(B) = D,$$
$$f_2(A, U) = E, \quad f_2(A, V) = F, \quad f_2(A, W) = E,$$
$$f_2(B, X) = E,$$

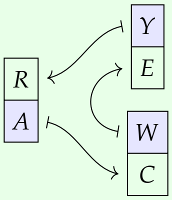
$$f^\sharp(A,U,Y)=S,\quad f^\sharp(A,U,Z)=R,\quad f^\sharp(A,W,Y)=R,\quad f^\sharp(A,W,Z)=R$$
$$f^\sharp(B,X,Y)=T,\quad f^\sharp(B,X,Z)=T$$

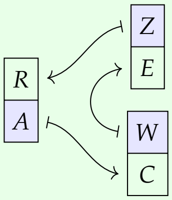
Box-notation uses a slightly different organization to represent the same data:

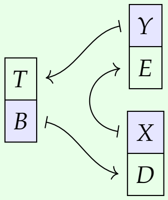


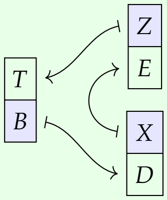












Example 5.34 (Using (5.29) to denote positions and directions in a composite). Suppose given polynomials $p_1, \ldots, p_k$. Recall from Exercise 4.7 that a position in their composite as a map

$$i\colon y^1 \to p_1 \triangleleft \cdots \triangleleft p_k.$$

We can denote i in the notation (5.29) 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

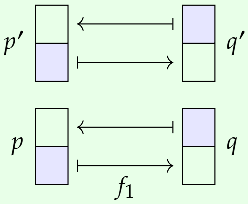
$$(p_1 \triangleleft \cdots \triangleleft p_k)[(i_1, \ldots, i_k)] \cong \sum_{d_1 \in p_1[i_1]} \sum_{d_2 \in p_2[i_2(d_1)]} \cdots \sum_{d_k \in p_k[i_k(d_1, \ldots, d_{k-1})]} 1$$

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 $f \in r[k(d, e)]$.

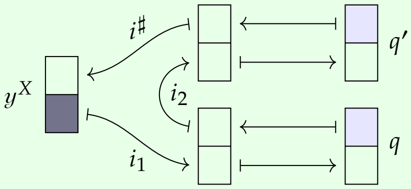
Exercise 5.35 (Solution here). Suppose $A_1, \ldots, A_k$ are sets and $p_i := A_i y$ for each i. Use the notation of Example 5.34 to give the set of positions in $p := p_1 \triangleleft \cdots \triangleleft p_k$. ◊

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 5.36 ($\triangleleft$ 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 5.37 (Solution here). 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. ◊

5.4 Mathematical aspects of $\triangleleft$

We now want to get at more subtle aspects of $p \triangleleft q$. We begin with the following.

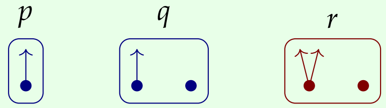
Proposition 5.38 (Left distributivity of $\triangleleft$)**.** For any polynomial r, the post-compose-with-r functor $(- \triangleleft r)\colon \mathbf{Poly} \to \mathbf{Poly}$ commutes—up to natural isomorphism—with addition and multiplication:

$$(p+q) \triangleleft r \cong (p \triangleleft r) + (q \triangleleft r) \qquad \text{and} \qquad pq \triangleleft r \cong (p \triangleleft r)(q \triangleleft r).$$

Proof. Formally, this just comes down to the fact that coproducts and products of functors $\mathbf{Set} \to \mathbf{Set}$ are computed pointwise and $\mathbf{Poly}$ 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 5.39. □

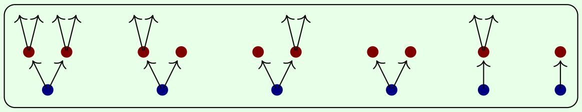
Exercise 5.39 (Solution here). Prove Proposition 5.38 in terms of the formula for $\triangleleft$ given in Eq. (5.5). ◊

Example 5.40 (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:



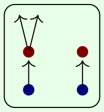
Then $pq \cong y^2 + y$ and we can draw $pq \triangleleft r$ as follows:

$(p \times q) \triangleleft r \cong$

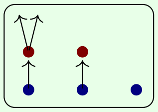


Or we can compute $p \triangleleft r$ and $q \triangleleft r$ seperately:

$p \triangleleft r \cong$

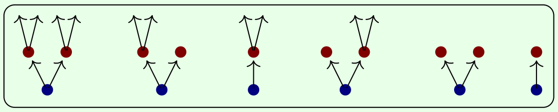


$q \triangleleft r \cong$



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 5.41 (Solution here). Follow Example 5.40 with + in place of ×: use pictures to give an intuitive understanding of the left-distributivity $(p+q) \triangleleft r \cong (p \triangleleft r) + (q \triangleleft r)$. ◊

Exercise 5.42 (Solution here). Show that the distributivities of Proposition 5.38 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 5.43. The following categories are connected:

$$\boxed{\bullet}\qquad \boxed{\bullet \rightrightarrows \bullet}\qquad \boxed{\bullet \to \bullet \leftarrow \bullet}\qquad \boxed{\bullet \leftarrow \bullet \leftarrow \bullet \leftarrow \cdots}$$

In particular, equalizers, pullbacks, and directed limits are examples of connected limits.

The following categories are *not connected*:

$$\boxed{}\qquad \boxed{\bullet \quad \bullet}\qquad \boxed{\bullet \quad \bullet \to \bullet}$$

In particular, terminal objects and products are *not* examples of connected limits.

Theorem 5.44 (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 5.38: limits of functors $\mathbf{Set} \to \mathbf{Set}$ are computed pointwise and $\mathbf{Poly}$ 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 $\mathbf{Set}$. See [GK12, Proposition 1.6] for details. □

Exercise 5.45 (Solution here). Use Theorem 5.44 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 5.44.
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 5.46. 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 5.47. For any polynomials p, p', q, q' there are natural maps

$$(p \triangleleft p') + (q \triangleleft q') \to (p + q) \triangleleft (p' + q') \tag{5.48}$$

$$(p \triangleleft p') \otimes (q \triangleleft q') \to (p \otimes pq) \triangleleft (p' \otimes q') \tag{5.49}$$

$$(p \triangleleft p') \times (q \triangleleft q') \leftarrow (p \times q) \triangleleft (p' \times q') \tag{5.50}$$

making $(+, \triangleleft)$ and $(\otimes, \triangleleft)$ duoidal structures and $(\times, \triangleleft)$ op-duoidal.

Proof. For (5.48) 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 (5.50) is similar.

It remains to give a map (5.50).** □

Proposition 5.51 ($\triangleleft$ **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

$$\begin{tikzcd}
pqA \arrow[rr] \arrow[dr] \arrow[dd] & & pqB \arrow[dr] \arrow[dd] & \\
& pq'A \arrow[rr] \arrow[dd] & & pq'B \arrow[dd] \\
pq'A \arrow[rr] \arrow[dr] & & pq'B \arrow[dr] & \\
& p'q'A \arrow[rr] & & p'q'B
\end{tikzcd}$$

are pullbacks by Proposition 4.55 and Theorem 5.44. Hence the diagonal is too by standard properties of pullbacks. □

Exercise 5.52 (Solution here).

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. ◊

5.5 Exercise solutions

Solution to Exercise 5.6.

We are given $p := y^2 + y^1$ and $q := y^3 + 1$.

1. By standard polynomial multiplication, we have that $y^2 \triangleleft q \cong q^2 \cong y^6 + 2y^3 + 1$.
2. We have that $y^1 \triangleleft q \cong q \cong y^3 + 1$.
3. Combining the previous parts, we have that $(y^2 + y^1) \triangleleft q \cong q^2 + q^1 \cong y^6 + 3y^3 + 2$.
4. Since $p[1] \cong 2$ and $q(1) \cong 2$, there are $2^2 = 4$ functions $p[1] \to q(1)$.
5. When $j_1\colon p[1] \to q(1)$ is one of the two possible bijections, we have that
$$\sum_{d \in p[1]} q[j_1(d)] \cong q[1] + q[2] \cong 3 + 0 \cong 3.$$
When $j_1\colon p[1] \to q(1)$ sends everything to $1 \in q(1)$, we have that
$$\sum_{d \in p[1]} q[j_1(d)] \cong q[1] + q[1] \cong 3 + 3 \cong 6.$$
Finally, when $j_1\colon p[1] \to q(1)$ sends everything to $2 \in q(1)$, we have that
$$\sum_{d \in p[1]} q[j_1(d)] \cong q[2] + q[2] \cong 0 + 0 \cong 0.$$
6. Since $p[2] \cong 1$ and $q(1) \cong 2$, there are $2^1 = 2$ functions $p[2] \to q(1)$.
7. When $j_2\colon p[2] \to q(1)$ maps to $1 \in q(1)$, we have that $\sum_{d \in p[2]} q[j_2(d)] \cong q[1] \cong 3$, and when $j_2\colon p[2] \to q(1)$ maps to $2 \in q(1)$, we have that $\sum_{d \in p[2]} q[j_2(d)] \cong q[2] \cong 0$.
8. From the previous parts, we have that
$$\sum_{i \in p(1)} \sum_{j_i\colon p[i] \to q(1)} y^{\sum_{d \in p[i]} q[j_i(d)]} \cong (2y^3 + y^6 + y^0) + (y^3 + y^0) \cong y^6 + 3y^3 + 2,$$
which agrees with what $p \triangleleft q$ should be.

Solution to Exercise 5.7.

1. Given representable polynomials $p := y^A$ and $q := y^B$, we have that $p \triangleleft q \cong \left(y^B\right)^A \cong y^{AB}$, which is also representable.
2. Given linear polynomials $p := Ay$ and $q := By$, we have that $p \triangleleft q \cong A(By) \cong ABy$, which is also linear.
3. Given constant polynomials $p := A$ and $q := B$, we have that $p \triangleleft q \cong A$, which is also constant (see also Exercise 5.17).

Solution to Exercise 5.11.

We have p, q as in (5.8) and $r := y^2 + 1$.

1. **
2.
3.
4.

5.

Solution to Exercise 5.13.

Here A and B are sets and p is a polynomial.

1. The isomorphism exists: we have that $(Ay) \otimes (By) \cong ABy \cong (Ay) \triangleleft (By)$.
2. The isomorphism exists: we have that $y^A \otimes y^B \cong y^{AB} \cong y^A \triangleleft y^B$.
3. The isomorphism does not necessarily exist: while $A \otimes B \cong AB$, we have that $A \triangleleft B \cong A$.
4. The isomorphism exists: we have that $Ay \otimes p \cong \sum_{i \in p(1)} Ay \otimes y^{p[i]} \cong \sum_{i \in p(1)} Ay^{p[i]} \cong Ap \cong Ay \triangleleft p$.
5. The isomorphism does not necessarily exist: if, say, $p = B$, then $y^A \otimes B \cong B$, while $y^A \triangleleft B \cong B^A$.
6. The isomorphism does not necessarily exist: if, say, $p = y^B$, then $y^B \otimes Ay \cong Ay^B$, while $y^B \otimes Ay \cong (Ay)^B \cong A^B y^B$.
7. The isomorphism exists: we have that $p \otimes y^A \cong \sum_{i \in p(1)} y^{p[i]} \otimes y^A \cong \sum_{i \in p(1)} y^{Ap[i]} \cong p \triangleleft y^A$.

Solution to Exercise 5.17.

1. **

Below, X is a set and p is a polynomial.

2. A constant functor composed with any functor is still the same constant functor, so $X \triangleleft p \cong X$. We can also verify this using Eq. (5.5):

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

3. When viewed as functors, it is easy to see that $p \triangleleft X \cong p(X)$. We can also verify this using Eq. (5.5):

$$p \triangleleft X \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in X} \prod_{e \in \varnothing} y \cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in X} 1 \cong \sum_{i \in p(1)} \prod_{d \in p[i]} X \cong \sum_{i \in p(1)} X^{p[i]} \cong p(X).$$

Solution to Exercise 5.21.

**

Solution to Exercise 5.23.

1. The polynomial y is the identity functor on **Set**.
2. Composing any functor with the identity functor yields the original functor, so $p \triangleleft y \cong p$.
3. **

Solution to Exercise 5.39.

Given $p, q, r \in$ **Poly**, recall that the positions of the sum $p+q$ form the set $p(1)+q(1)$, while the directions at each position $i \in p(1) + q(1)$ correspond to the directions in $p[i]$ if $i \in p(1)$ and the directions in $q[i]$ if $i \in q(1)$. So by Eq. (5.5), we have that

$$\begin{aligned}(p+q) \triangleleft r &\cong \sum_{i \in p(1)+q(1)} \prod_{d \in (p+q)[i]} \sum_{j \in r(1)} \prod_{e \in r[j]} y \\ &\cong \sum_{i \in p(1)} \prod_{d \in p[i]} \sum_{j \in r(1)} \prod_{e \in r[j]} y + \sum_{i \in q(1)} \prod_{d \in q[i]} \sum_{j \in r(1)} \prod_{e \in r[j]} y \\ &\cong p \triangleleft r + q \triangleleft r.\end{aligned}$$

We can also recall that the positions of the product pq form the set $p(1) \times q(1)$, while the directions at each position $(i, i') \in p(1) \times q(1)$ correspond to the directions in $p[i] + q[i']$. So by Eq. (5.5), we have that

$$\begin{aligned}
(pq) \triangleleft r &\cong \sum_{(i,i')\in p(1)\times q(1)} \prod_{d\in p[i]+q[i']} \sum_{j\in r(1)} \prod_{e\in r[j]} y \\
&\cong \sum_{(i,i')\in p(1)\times q(1)} \left(\prod_{d\in p[i]} \sum_{j\in r(1)} \prod_{e\in r[j]} y\right)\left(\prod_{d\in q[i']} \sum_{j\in r(1)} \prod_{e\in r[j]} y\right) \\
&\cong \left(\sum_{i\in p(1)} \prod_{d\in p[i]} \sum_{j\in r(1)} \prod_{e\in r[j]} y\right)\left(\sum_{i'\in q(1)} \prod_{d\in q[i']} \sum_{j\in r(1)} \prod_{e\in r[j]} y\right) \\
&\cong (p \triangleleft r)(q \triangleleft r).
\end{aligned}$$

Solution to Exercise 5.42.

1. Let $p \coloneqq y+1, q \coloneqq 1$, and $r \coloneqq 0$. Then $p \triangleleft (qr) \cong (y+1) \triangleleft 0 \cong 1$, while $(p \triangleleft q)(p \triangleleft r) \cong ((y+1) \triangleleft 1)((y+1) \triangleleft 0) \cong 2 \times 1 \cong 2$.
2. Again let $p \coloneqq y+1, q \coloneqq 1$, and $r \coloneqq 0$. Then $p \triangleleft (q+r) \cong (y+1) \triangleleft 1 \cong 2$, while $(p \triangleleft q) + (p \triangleleft r) \cong ((y+1) \triangleleft 1) + ((y+1) \triangleleft 0) \cong 2 + 1 \cong 3$.

Solution to Exercise 5.45.

1. Given a polynomial functor $p\colon \mathbf{Set} \to \mathbf{Set}$, we wish to show that p preserves connected limits of sets; that is, for a connected category J and a functor $X\colon J \to \mathbf{Set}$, we have
$$p\left(\lim_{j\in J} X_j\right) \cong \lim_{j\in J} p(X_j).$$
But we can identify **Set** with the full subcategory of constant functors in **Poly** and instead view X as a functor into **Poly**. Then by Exercise 5.17 #3, the left hand side of the isomorphism we seek is isomorphic to $p \triangleleft \left(\lim_{j\in J} X_j\right)$, while the right hand side is isomorphic to $\lim_{j\in J}\left(p \triangleleft X_j\right)$. These are isomorphic by Theorem 5.44.
2. Given $p, q, r \in \mathbf{Poly}$, we wish to show that $p \triangleleft (qr) \cong (p \triangleleft q) \times_{(p\triangleleft 1)} (p \triangleleft r)$. As 1 is terminal in **Poly**, the product qr can also be written as the pullback $q \times_1 r$. While products are not connected limits, pullbacks are, so by Theorem 5.44, they are preserved by precomposition with p. Hence the desired isomorphism follows.
3. As in the previous part, we can write pq as the pullback $p \times_1 q$. By Theorem 5.44, this connected limit is preserved by composition with r, so $(pq)\triangleleft r \cong (p\triangleleft r)\times_{1\triangleleft r}(q\triangleleft r) \cong (p\triangleleft r)\times_1(q\triangleleft r) \cong (p\triangleleft r)(q\triangleleft r)$. (Here we use the fact that $1 \triangleleft r \cong 1$ by Exercise 5.17 #3.)
4. Given a set A and polynomials p, q, the left-distributivity of $\triangleleft$ over products implies that $(Ap)\triangleleft q \cong (A \triangleleft q)(p \triangleleft q)$, while Exercise 5.17 #3 implies that $A \triangleleft q \cong A$. So $(Ap) \triangleleft q \cong A(p \triangleleft q)$.

Solution to Exercise 5.52.

1. **
2. **

Chapter 6

Polynomial comonoids

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?

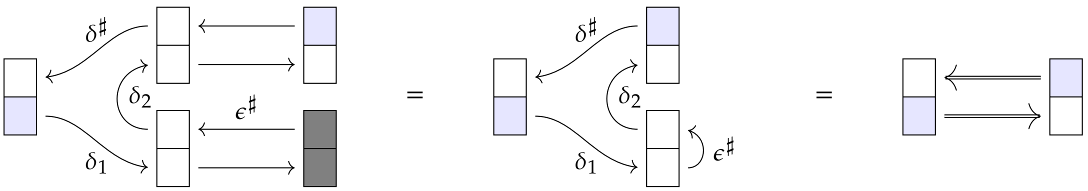
The most surprising aspects of **Poly** really begin with its comonoids. In 2018, researchers Daniel Ahman and Tarmo Uustalu showed that comonoids in (**Poly**, y, $\triangleleft$) can be identified with categories. Every category in the usual sense is a comonoid in **Poly** and every comonoid in **Poly** is a category. We find this revelation to be truly shocking, and it suggests some very different ways to think about categories. Let's go through it.

Definition 6.1 (Comonoid)**.** A *comonoid* in a monoidal category $(\mathcal{C}, y, \triangleleft)$ 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:

$$\begin{tikzcd}
y \triangleleft c \arrow[r, equal] & c \arrow[r, equal] \arrow[d, "\delta"] & c \triangleleft y \\
& c \triangleleft c \arrow[ul, "\epsilon \triangleleft c"] \arrow[ur, "c \triangleleft \epsilon"'] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
c \arrow[r, "\delta"] \arrow[d, "\delta"'] & c \triangleleft c \arrow[d, "c \triangleleft \delta"] \\
c \triangleleft c \arrow[r, "\delta \triangleleft c"'] & c \triangleleft c \triangleleft c
\end{tikzcd}
\qquad (6.2)$$

We refer to a comonoid $P \coloneqq (p, \epsilon, \delta)$ in (**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 6.3 (Solution here).

1. Draw the other unitality equation.
2. Draw the associativity equation. ◊

Example 6.4 (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* $p\colon \mathbf{Set} \to \mathbf{Set}$, 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.

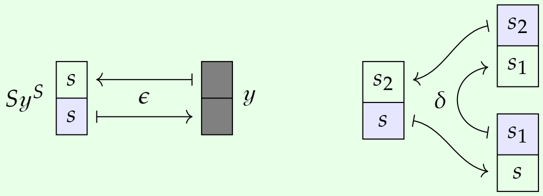
Example 6.5 (The state comonad Sy^S). Let S be a set, and consider the polynomial $p := Sy^S$. It has a canonical comonoid structure—often called the *state* comonad—as we discussed in **??**, page **??**. To say it in the current language, we first need to give maps $\epsilon\colon p \to y$ and $\delta\colon p \to p \triangleleft p$. By Eqs. (5.5) and (5.22), 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

$$s \overset{\epsilon'}{\mapsto} s \qquad\qquad s \overset{\delta'}{\mapsto} (s' := s, s_1 \mapsto (s_1' := s_1, s_2 \mapsto s_2)). \tag{6.6}$$

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.

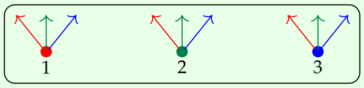


(6.7)

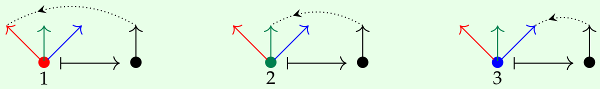
Exercise 6.8 (Solution here). Let $p := Sy^S$. For any $n \in \mathbb{N}$, write out the morphism of polynomials $\delta^n \colon p \to p^{\triangleleft(n+1)}$ either set-theoretically or in terms of poly-boxes as in Example 6.5 ◊

Example 6.9 (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 $\{\bullet, \bullet, \bullet\}y^{\{\bullet,\bullet,\bullet\}}$:

$$3y^3 =$$

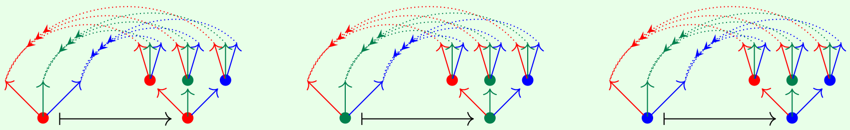


The map $\epsilon \colon Sy^S \to y$ can be drawn as follows:



It picks out one direction at each position, namely the one of the same color.

The map $\delta \colon Sy^S \to (Sy^S)^{\triangleleft 2}$ can be drawn as follows:



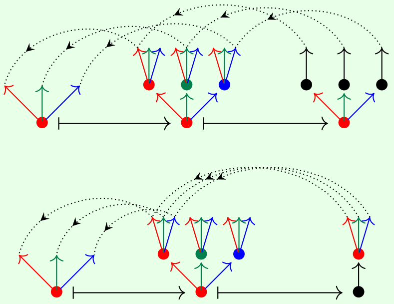
(6.10)

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: (6.10) 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 (6.2). 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 6.11.

---

[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 4.51.

Exercise 6.11 (Solution here). 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 6.9.
4. Are they equal? ◊

Exercise 6.12 (Monoid actions; solution here). Suppose that $(M, e, *)$ is a monoid, S is a set, and $\alpha\colon M \times S \to S$ is an M-action.

1. Show that Sy^M forms a comonoid.
2. Show that the projection $Sy^M \to y^M$ is a comonoid homomorphism.
3. M always acts on itself by multiplication. Is the associated comonoid structure on My^M the same or different from the one coming from Example 6.5? ◊

6.1 Speeding up dynamical systems

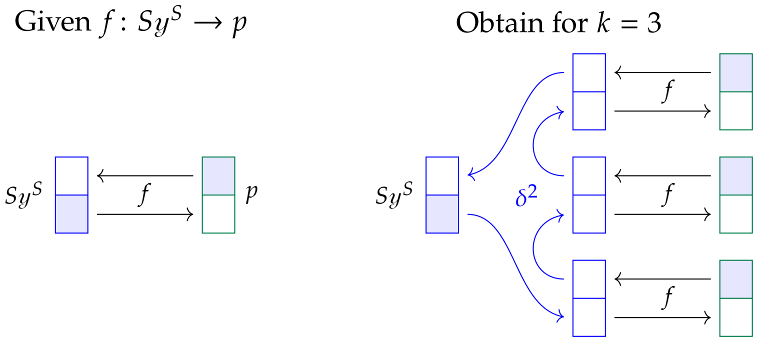
Suppose we have a dynamical system $f\colon Sy^S \to p$, and we want to make it go k-times faster. That is, in every moment, we want it to process k-many inputs, rather than one.

Since Sy^S has the structure of a comonoid, we know that for every $k \in \mathbb{N}$ we have a map $\delta^{k-1}\colon Sy^S \to (Sy^S)^{\triangleleft k}$ by Example 6.4. But we also have maps $f^{\triangleleft k}\colon (Sy^S)^{\triangleleft k} \to p^{\triangleleft k}$

because $\triangleleft$ is a monoidal product. Thus we can form the composite

$$\begin{tikzcd} Sy^S \arrow[r, "\delta^{k-1}"] \arrow[rr, bend right, "\mathrm{Spdup}_k(f)"'] & (Sy^S)^{\triangleleft k} \arrow[r, "f^{\triangleleft k}"] & p^{\triangleleft k} \end{tikzcd} \tag{6.13}$$

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:

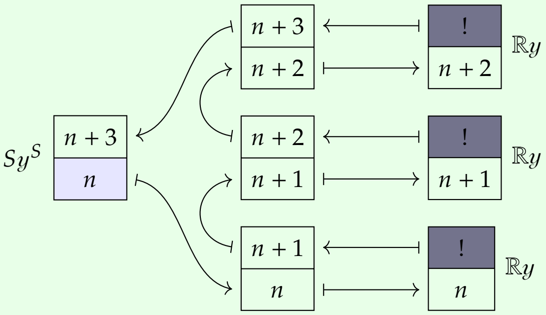


Example 6.14. Let $p := \mathbb{R}y^1$, let $S := \mathbb{N}$, and let $\binom{f^\sharp}{f_1}\colon Sy^S \to p$ be given by $f_1(n) := n$ and $f^\sharp(n, 1) := n + 1$. What is this speedup map $\mathrm{Spdup}_k(f)$? First of all, its type is

$$\mathrm{Spdup}_k(f)\colon Sy^S \to \mathbb{R}^k y,$$

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:

$$(0, 1, 2), (3, 4, 5), (6, 7, 8), (9, 10, 11).$$

6.2 Other comonoids

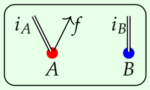
Once you know that these all important Sy^S-things are comonoids in **Poly**, it's interesting to ask "what are all the comonoids in **Poly**?" Let's discuss another one before answering the question in generality.

Example 6.15 (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.

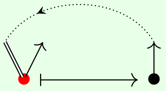
Define $w := \{A\}y^{\{i_A, f\}} + \{B\}y^{\{i_B\}}$; it is clearly isomorphic to $y^2 + y$, but its notation is meant to remind the reader of the walking arrow category

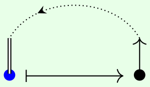
$$\mathcal{W} := \boxed{A \xrightarrow{f} B}$$

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 $w \cong y^2 + y$:

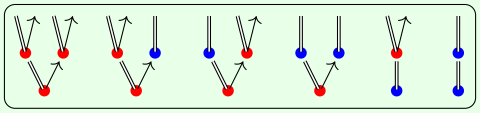
$w :=$ 

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 $\epsilon^\sharp(A) := i_A$ and $\epsilon^\sharp(B) := i_B$:

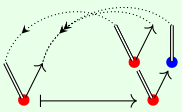


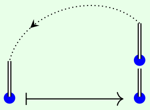


Now we need a map of polynomials $\delta\colon w \to w \triangleleft w$. Let's draw out $w \triangleleft w$ to see what it looks like.

$w \triangleleft w =$ 

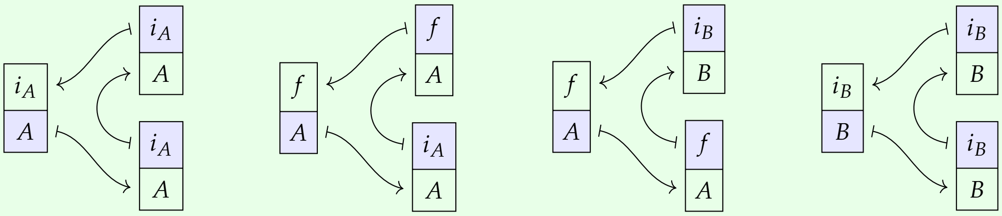
The map δ is going to tell us both about codomains and composition. Here it is:



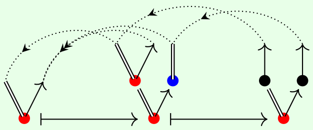


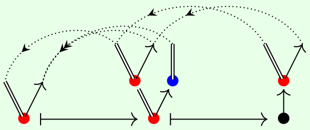
The on-positions map selects, for each position (either A or B) the two-level tree starting at that position and having the correct codomains: the identity arrow on A points to the corolla for A; the f map points to the corolla for B; and the identity arrow on B

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 (6.2) 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 6.16 (Solution here). Write out the data (c, ϵ, δ) for the comonoid corresponding to the category

$$\boxed{B \xleftarrow{f} A \xrightarrow{g} C}$$

For this exercise, you are not being asked to check the unitality or associativity conditions. ◊

Exercise 6.17 (Solution here). Show that if A is a set and $p \coloneqq Ay$ is the associated linear polynomial, then there exists a unique comonoid structure on p. ◊

Example 6.18 (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 7.13.

6.3 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 6.19 (Cofunctor)**.** Let $\mathcal{C}$ be a category with object set C_0, morphism set C_1, $\mathrm{dom}, \mathrm{cod}\colon C_1 \to C_0$ the domain and codomain,[a] and similarly for $\mathcal{D}$. A *cofunctor* $F\colon \mathcal{C} \nrightarrow \mathcal{D}$ consists of

1. a function $F\colon C_0 \to D_0$ *on objects* and
2. a function $F^\sharp\colon C_0 \times_{D_0} D_1 \to C_1$ *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 $\mathrm{dom}\colon C_1 \to C_0$ in the sense that an unnamed map $C_1 \to C_0$ will be assumed to be dom. For example, in the map $F^\sharp$, the pullback $C_0 \times_{D_0} D_1$ is of the diagram $C_0 \xrightarrow{F} D_0 \xleftarrow{\mathrm{dom}} D_1$.

The cofunctor laws can be written in commutative diagram form as follows:

$$\begin{tikzcd}
C_0 \times_{D_0} D_0 \arrow[r, "\cong"] \arrow[d, "\mathrm{id}_D"'] \arrow[dr, phantom, "\text{(i)}"] & C_0 \arrow[d, "\mathrm{id}_C"] \\
C_0 \times_{D_0} D_1 \arrow[r, "F^\sharp"'] & C_1
\end{tikzcd}
\qquad
\begin{tikzcd}
C_0 \times_{D_0} D_1 \arrow[r, "F^\sharp"] \arrow[d, "\pi_2"'] & C_1 \arrow[r, "\mathrm{cod}"] \arrow[d, phantom, "\text{(ii)}"] & C_0 \arrow[d, "F_0"] \\
D_1 \arrow[rr, "\mathrm{cod}"'] & & D_0
\end{tikzcd}$$

$$\begin{tikzcd}
C_0 \times_{D_0} D_1 \times_{D_0} D_1 \arrow[r, "\fatsemi_D"] \arrow[d, "F^\sharp"'] & C_0 \times_{D_0} D_1 \arrow[r, "F^\sharp"] \arrow[d, phantom, "\text{(iii)}"] & C_1 \\
C_1 \times_{D_0} D_1 \arrow[r, "\cong"'] & C_1 \times_{C_0} C_0 \times_{D_0} D_1 \arrow[r, "F^\sharp"'] & C_1 \times_{C_0} C_1 \arrow[u, "\fatsemi_C"']
\end{tikzcd}$$

Example 6.20 (Admissible sections). Consider the monoid $\mathcal{N} := (\mathbb{N}, 0, +)$, considered as a category with one object. For any category $\mathcal{C}$, a cofunctor $\varphi\colon \mathcal{C} \nrightarrow \mathcal{N}$ is called an *admissible section* [**Aguiar-thesis**]. We'll have more to say about these in Theorem 7.43, but our goal here is simply to unpack the definition.

To specify φ, we first say what it does on objects, but this is trivial: there is only one object in $\mathcal{N}$, so each object of $\mathcal{C}$ is sent to it. So the content of φ is all found in $\varphi^\sharp$, which assigns to each object $i \in \mathcal{C}$ and natural number $n \in \mathbb{N}$ a morphism $\varphi^\sharp(i, n)$ emanating from i. That seems like a lot of data, but we still have two laws to pare it down:

$$\varphi^\sharp(i,0)=i \qquad \text{and} \qquad \varphi^\sharp(i,n+n')=\varphi^\sharp(\varphi^\sharp(i,n),n').$$

Every natural number n is a sum of 1's, so if we denote $\varphi^\sharp(i,1)$ by $\phi(i)$, we in fact have

$$\varphi^\sharp(i,n)=\phi^{\circ n}i.$$

That is, $\varphi^\sharp(i,n)$ is just the n-fold application of the one-step case, ϕ.

Thus an admissible section of $\mathcal{C}$ is given by choosing, for each object $i\in\mathcal{C}$, a morphism $\phi(i)\colon i\to i'$ emanating from i.

Exercise 6.21 (Solution here). How many admissible sections does the category $\boxed{\bullet\to\bullet}$ have? ◊

Exercise 6.22 (Solution here). Let $\mathcal{Z}\coloneqq(\mathbb{Z},0,+)$ denote the monoid of integers and let $\mathcal{N}$ be that of natural numbers as above.

1. For a category $\mathcal{C}$, describe the data of a cofunctor $\mathcal{C}\nrightarrow\mathcal{Z}$.
2. What would you say is the canonical cofunctor $\mathcal{Z}\nrightarrow\mathcal{N}$?
3. Thinking of an admissible section $\mathcal{C}\nrightarrow\mathcal{N}$ as a policy, almost like a *physical law*, suppose it factors through $\mathcal{Z}$. Would you say that this lets you "run the law backwards"? ◊

Example 6.23 (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

$$x^{+0}=x \qquad \text{and} \qquad x^{+t_1+t_2}=(x^{+t_1})^{+t_2}. \tag{6.24}$$

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 (6.24) 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

$$F^\sharp(x^{+t_1}, t_2) + F^\sharp(x, t_1) = (x^{+t_1})^{+t_2} - x^{+t_1} + x^{+t_1} - x = x^{+t_1+t_2} - x = F^\sharp(x, t_1 + t_2).$$

Exercise 6.25 (Solution here).

1. Suppose that M, N are monoids (each is a category with one object). Are cofunctors between them related to monoid homomomorphisms? If so, how?
2. Suppose $\mathcal{C}$ and $\mathcal{D}$ are categories and $F\colon \mathcal{C} \nrightarrow \mathcal{D}$ is a cofunctor. Does there necessarily exist a cofunctor $\mathcal{C}^{\mathrm{op}} \nrightarrow \mathcal{D}^{\mathrm{op}}$ that acts the same as F on objects? ◊

Theorem 6.26 (Ahman-Uustalu)**.** There is an equivalence of categories

$$\mathbf{Comon}(\mathbf{Poly}) \cong \mathbf{Cat}^\sharp.$$

Proof. This will be proved as Proposition 7.20 and Theorem 7.1. □

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 $\{f\colon i \to j \mid j \in \mathrm{Ob}(\mathcal{C})\}$ emanating from i in $\mathcal{C}$ are the directions $\mathfrak{c}[i]$ there.

Definition 6.27. Let $\mathcal{C}$ be a category. The *emanation polynomial for* $\mathcal{C}$ is the polynomial

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

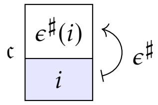
Exercise 6.28 (Solution here). What is the emanation polynomial for each of the following categories?

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 $(\mathbb{N}, \geq)$? ◊

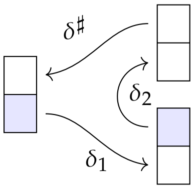
A category $\mathcal{C}$ is more than its emanation polynomial $\mathfrak{c}$, and a comonoid $(\mathfrak{c}, \epsilon, \delta)$ in **Poly** is more than its carrier polynomial $\mathfrak{c}$. The identities of $\mathcal{C}$ are all captured by the counit $\epsilon \colon \mathfrak{c} \to y$ and the codomain and composition information of $\mathcal{C}$ are all captured by the comultiplication map $\delta \colon \mathfrak{c} \to \mathfrak{c} \mathbin{\triangleleft} \mathfrak{c}$. Our goal is to make this clear so that we can justly proclaim:

Comonoids in **Poly** *are precisely categories!*

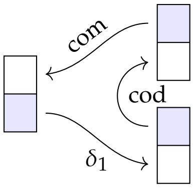
We want to understand how the counit ϵ and comultiplication δ in a comonoid $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ relate to identities, codomains, and composites in a category. We first use our work in Section 5.3 to get a better handle on ϵ and δ. For example, since $\epsilon \colon \mathfrak{c} \to y$ maps to the empty composite, we know by (5.31) 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 (5.32) that δ is of the form

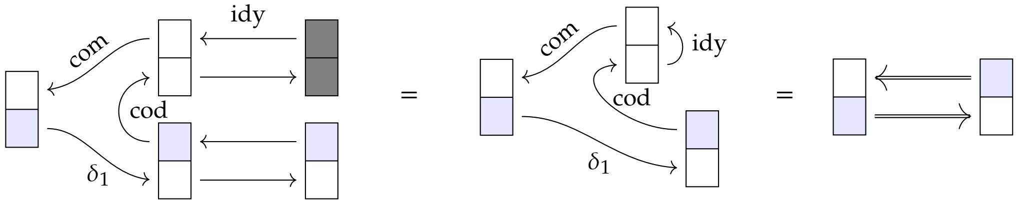


We've said that this secretly holds information about the codomains and composites for a category structure on $\mathfrak{c}$. How does that work? We will soon find that δ_1 is forced to be an identity, that δ_2 holds codomain information, and that $\delta^\sharp$ holds composite information. So we will use that notation here

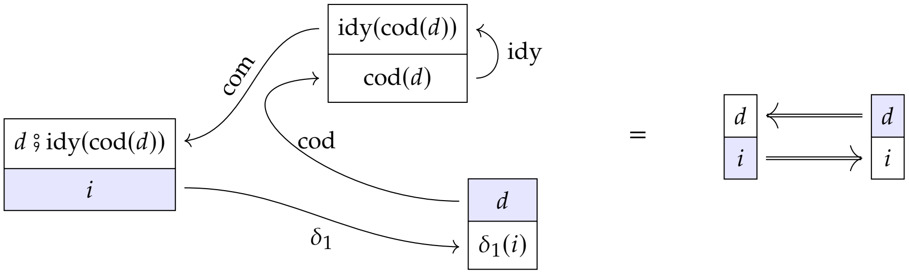


and our goal now is to see that the cod map really has something to do with codomains and that the com map really has something to do with composites, as advertised. What makes these true are the unitality and associativity equations required for $(\mathfrak{c}, \epsilon, \delta)$ to be a comonoid; see Definition 6.1.

To get started, we consider the first unitality equation from (6.2):

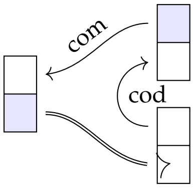


Let's add some arbitrary fillers $i \in \mathfrak{c}(1)$ and $d \in \mathfrak{c}[i]$ to the open slots, and hence obtain an equation:



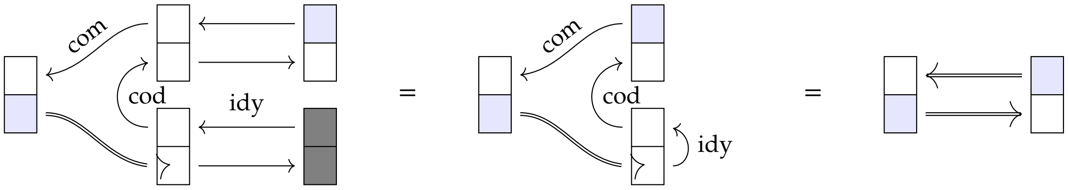
First it's saying that $\delta_1(i) = i$. This is great news; it means we can forget about δ_1, just as we said earlier. Second it's saying that $d \mathbin{⨟} \text{idy}(\text{cod}(d)) = d$. Unpacking, this means that composing a morphism d with the identity morphism on its codomain returns d. It's 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 $\delta \colon c \to c \triangleleft c$ with the information-lacking δ_1 (which the first unitality equation said was always identity) and replace it with a double arrow:

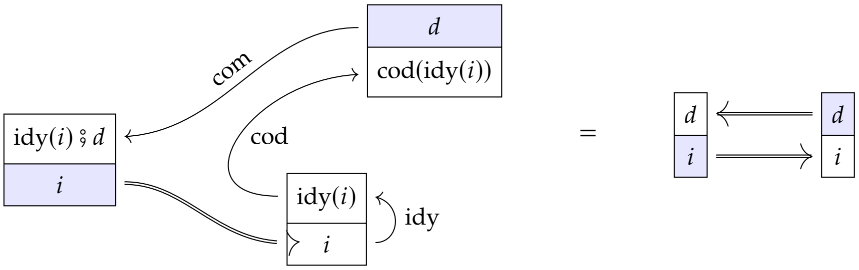


(6.29)

Now we can write the other unitality equation.

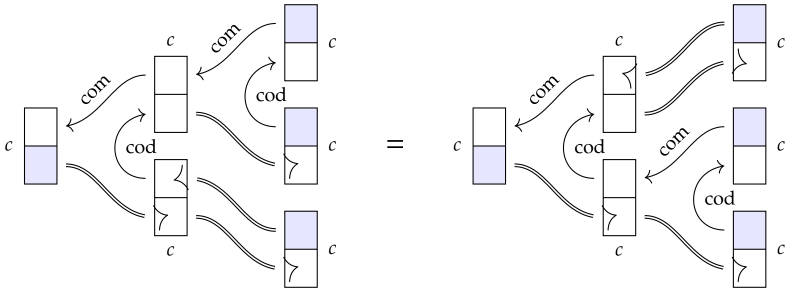


Let's add some arbitrary fillers $i \in c(1)$ and $d \in c[i]$ to get some equations:

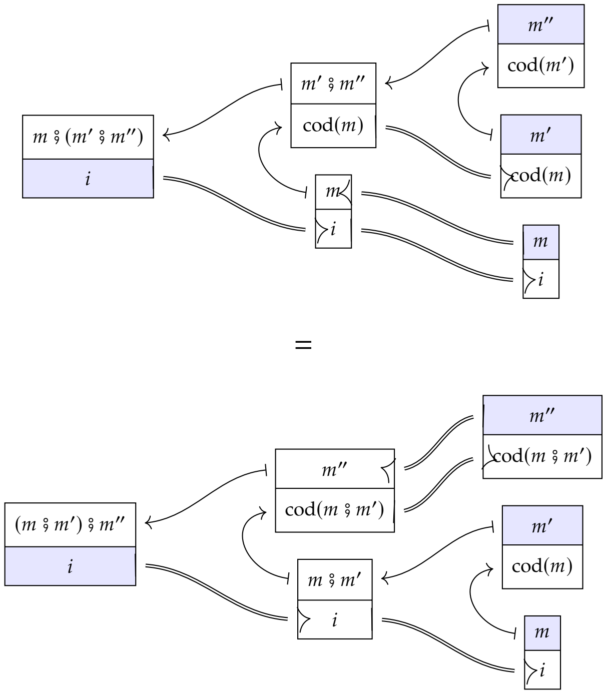


Ah, it's saying that it wants $\text{cod}(\text{idy}(i)) = i$, which makes sense—the codomain of the identity on i should be i—and that it wants $\text{idy}(i) ⨾ d = d$, i.e. that composing the identity on i with d should return d. We couldn't have said it better ourselves; thanks like-minded toad!

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 6.30 (Solution here). 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. ◊

6.4 Examples showing the correspondence between comonoids and categories

Example 6.31 (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 6.32 (Solution here). Finish Example 6.31 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 6.33 (Monoid action). Suppose that $(M, e, *)$ is a monoid, S is a set, and $\alpha \colon M \times S \to S$ is an action. There is an associated category $\mathcal{A}$ with emanation polynomial Sy^M. In other words, it has objects set S and for every $s \in S$ there is an outgoing morphism for each $m \in M$, namely $s \xrightarrow{m} \alpha(m, s)$.

Exercise 6.34 (Solution here). With notation as in Example 6.33,

1. For a given object $s \in \mathcal{A}$, what is the identity morphism?
2. What is the composite of two morphisms in $\mathcal{A}$? ◊

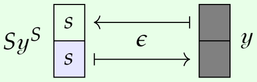
Example 6.35 (Cyclic lists). For any $n \in \mathbb{N}$, consider the monoid (group) $\mathbb{Z}/n$. As a functor $c_n \coloneqq y^{\mathbb{Z}/n}$ sends a set X to the set of length-n tuples in X. But the comonoid structure lets us think of these as cyclic lists. Indeed, $\epsilon\colon c_n \to y$ allows us to pick out the "current" element via the map $\epsilon \triangleleft X\colon c_n \triangleleft X \to X$, and δ lets us move around the list.

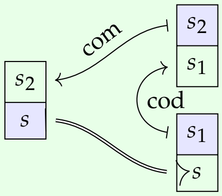
We will see later that comonoids are closed under coproducts, so $\sum_{n \in \mathbb{N}} c_n$ is also a comonoid.

Example 6.36 (What category is Sy^S?). The first comonoid we introduced, back in Example 6.5 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.

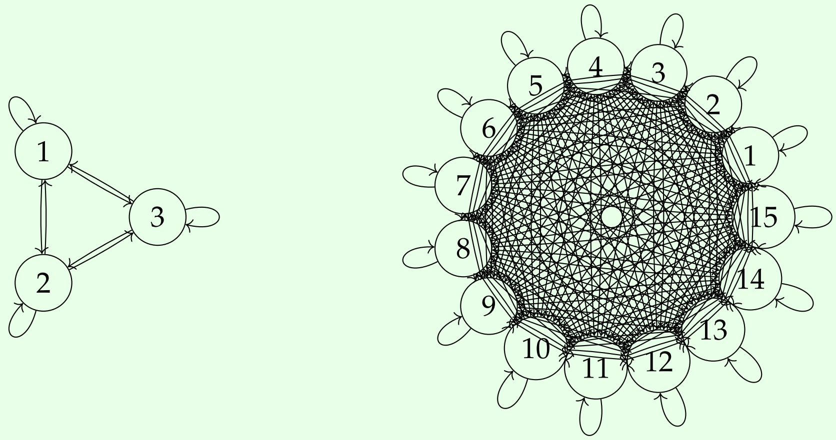
To calculate the codomains and compositions we examine the map $\delta\colon p \to p \triangleleft p$, which was given set-theoretically in (6.6) and in terms of poly-boxes in (6.7). We repeat it here for your convenience:





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 6.37 (Solution here). We showed in Exercise 6.17 that for any set A, the linear polynomial $p := Ay$ has a unique comonoid structure. What category does it correspond to? ◊

Definition 6.38. 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 $\deg(c) \cong \mathsf{1}$, 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 6.39 (Solution here).

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 comonoid structures on $y^2 + y$ there are?

◊

Exercise 6.40 (Solution here).

1. Find a category structure for the polynomial $y^{\mathsf{n}+1} + \mathsf{n}y$.
2. Would you call your category "star-shaped"? ◊

Exercise 6.41 (Solution here). Let S be a set. Is there any comonoid structure on Sy^S other than that of the state category? ◊

6.5 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 $\mathcal{C} \to \mathcal{D}$.

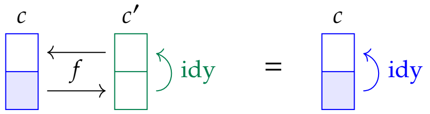
Cofunctors F are forward on objects, and backwards on morphisms. It's good to remember: Codomains are objects, so F preserves them going forwards; identities and composites are morphisms, so F preserves them going backwards.

Let's begin with a definition.

Definition 6.42 (Morphisms of comonoids)**.** Let $\mathscr{C} := (c, \epsilon, \delta)$ and $\mathscr{C}' := (c', \epsilon', \delta')$ be polynomial comonoids as in Definition 6.1. A *morphism* $\mathscr{C} \to \mathscr{C}'$ consists of a morphism $f\colon c \to c'$ of polynomials that commutes with the structure maps:

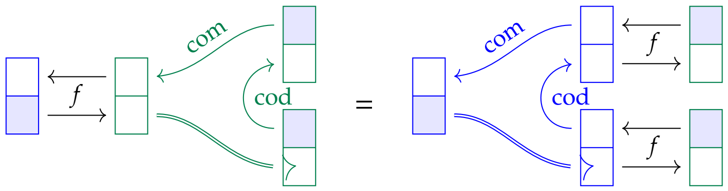
$$\begin{tikzcd}
c \arrow[r, "f"] \arrow[d, "\epsilon"'] & c' \arrow[d, "\epsilon'"] & & c \arrow[r, "f"] \arrow[d, "\delta"'] & c' \arrow[d, "\delta'"] \\
y \arrow[r, equal] & y & & c \triangleleft c \arrow[r, "f \triangleleft f"'] & c' \triangleleft c'
\end{tikzcd} \tag{6.43}$$

Let's see the two laws of comonoid morphisms using poly-boxes. First the counit law:



(6.44)

Then the comultiplication law:



(6.45)

If we fill in Eq. (6.44) 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 6.19. If we fill in Eq. (6.45) 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

$$\mathrm{cod}(m) = f_1\big(\mathrm{cod}\left(f^\sharp(i, m)\right)\big)$$
$$f^\sharp(i, \mathrm{com}(m, m')) = \mathrm{com}(f^\sharp(i, m), f^\sharp(\mathrm{cod}(f^\sharp(i, m)), m'))$$

and these are the second and third laws of Definition 6.19.

Exercise 6.46 (Solution here). Summarize the proof of Theorem 6.26, developed above. You may cite anything written in the text so far. ◊

Proposition 6.47. 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 $F^\sharp_c(g)$.

Proof. With $d := F(c)$, $d' := F(c')$, and g' the inverse of g, we have

$$\begin{aligned} \mathrm{id}_c &= F^\sharp_c(\mathrm{id}_d) \\ &= F^\sharp_c(g \mathbin{⨾} g') \\ &= F^\sharp_c(g) \mathbin{⨾} F^\sharp_{c'}(g') \end{aligned}$$

Thus $F^\sharp_c(g)$ is a section of $F^\sharp_{c'}(g')$. The opposite is true similarly, completing the proof. □

6.5.1 Examples of cofunctors

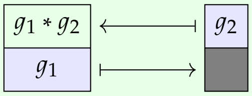
We saw in Theorem 6.26, summarized in Exercise 6.46, that cofunctors $\mathcal{C} \nrightarrow \mathcal{D}$ are the same thing as morphisms of comonoids $\mathscr{C} \to \mathscr{D}$, so we elide the difference. The question we're interested in now is: how do we think about cofunctors? What is a map 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 6.48. 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 6.49 (Solution here). Does the idea of Example 6.48 work when G is merely a monoid, or does something go subtly wrong somehow? ◊

Proposition 6.50. 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 6.51. 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 6.52. 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 $|\cdot| : \mathcal{C} \nrightarrow \mathbb{R}y^{\mathbb{R}}$ a cofunctor. It assigns to every object c both a real number $|c| \in \mathbb{R}$ and a choice of emanating morphism $|c|^{\sharp}(r) : c \to c_r$ 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 6.53 (Solution here).

1. Do you think a cofunctor $\mathcal{C} \nrightarrow \mathbb{R}y^{\mathbb{R}}$ as in Example 6.52 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? ◊

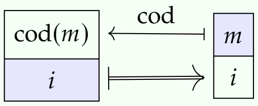
Exercise 6.54 (Solution here).

1. Over two discrete objects $\{A, B\}$, how many cofunctors

$$y^2 + y \cong \boxed{A \to B} \nrightarrow \boxed{A \rightrightarrows B} \cong y^3 + y$$

are there 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 6.55 (Solution here). 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 6.56 (Canonical cofunctors from state categories). Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid, where $\delta = (\text{id}, \text{cod}, \text{com})$ as in (6.29). For any position $i \in \mathfrak{c}(1)$, there is a cofunctor

$$(\text{cod}, \text{com}) \colon \mathfrak{c}[i]y^{\mathfrak{c}[i]} \nrightarrow \mathfrak{c}.$$

That is, an object $f \in \mathfrak{c}[i]$ is also a morphism in $\mathcal{C}$ and we send it to its codomain $\text{cod}(f)$. A morphism in $\mathcal{C}$ emanating from $\text{cod}(f)$ is passed back to its composite with f.

Exercise 6.57 (Solution here). Suppose $\mathcal{C} = (\mathfrak{c}, \epsilon, \delta)$ is a comonoid.

1. Show that the map $(\text{cod}, \text{com}) \colon \mathfrak{c}[i]y^{\mathfrak{c}[i]} \nrightarrow \mathfrak{c}$ from Example 6.56 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 6.58 (Solution here). Suppose $\mathfrak{c}, \mathfrak{d}, \mathfrak{e}$ are polynomials, each with a comonoid structure, and that $f\colon \mathfrak{c} \to \mathfrak{d}$ and $g\colon \mathfrak{d} \to \mathfrak{e}$ are maps of polynomials.

1. If f and $f \mathbin{⨟} g$ are each cofunctors, is g automatically a cofunctor? If so, sketch a proof; if not sketch a counterexample.
2. If g and $f \mathbin{⨟} g$ are each cofunctors, is f automatically a cofunctor? If so, sketch a proof; if not sketch a counterexample. $\Diamond$

Exercise 6.59 (Solution here).

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? $\Diamond$

Exercise 6.60 (Solution here). 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? $\Diamond$

Exercise 6.61 (Solution here).

1. What is the finite set $\{\mathscr{Q}_1, \ldots, \mathscr{Q}_n\}$ of comonoids (defined up to isomorphism) for which the carrier polynomial is $y^2 + y$?
2. For each category $\mathscr{Q}_i$, describe how to imagine a cofunctor $\mathscr{C} \to \mathscr{Q}_i$ from an arbitrary category into it.
3. What cofunctors exist between the various $\mathscr{Q}_i$? $\Diamond$

Exercise 6.62 (Solution here). 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 6.47. $\Diamond$

Exercise 6.63 (Solution here).

1. Recall the star-shaped category $y^{\mathsf{n}+1} + \mathsf{n}y$ from Exercise 6.40. 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. ◊

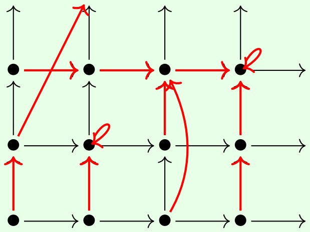
Example 6.64 (Objects aren't representable in $\mathbf{Cat}^\sharp$). For categories and ordinary functors, there is a category $\mathcal{T}$ that *represents objects*, in the sense that functors $\mathcal{T} \to \mathcal{C}$ are the same as objects in $\mathcal{C}$; indeed, take $\mathcal{T} = \boxed{\bullet}$ to be the terminal (one morphism) category.

This does not work for cofunctors, as we'll see in Exercise 6.65. 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 6.19, condition 2).

Exercise 6.65 (Solution here). We saw in Exercise 6.17 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 6.64 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 6.66 (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

$$\mathcal{C} \nrightarrow \mathcal{n}$$

where $\mathcal{n} = y^{\mathbb{N}}$ is the monoid of natural numbers under addition.

Exercise 6.67 (Solution here). At the end of Example 6.66 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 $F_c^\sharp(n)$ 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 7.44 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 6.68 (Continuous trajectories; solution here). 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 6.69 (Solution here). 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 6.70 (Solution here).

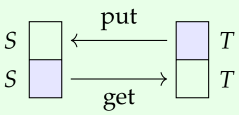
1. If two categories are isomorphic in **Cat**, does that imply they are isomorphic in $\mathbf{Cat}^\sharp$?
2. If so, prove it; if not, give a counterexample.
3. Is it true that for any two categories $\mathcal{C}, \mathcal{D}$, there is a bijection between the set of isomorphisms $\mathcal{C} \xrightarrow{\cong} \mathcal{D}$ 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. ◊

6.5.2 Very well behaved lenses

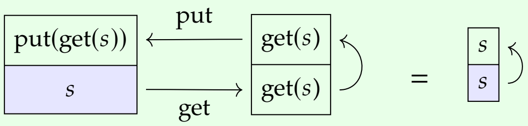
In the functional programming community, there is an important notion of very well-behaved lenses. These turn out to be precisely the cofunctors between state categories. Since state categories Sy^S play an important role in our theory, we take a bit of time to consider cofunctors between them.

Example 6.71 (Very well-behaved lenses). Recall from Example 6.5 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 $f\colon Sy^S \to Ty^T$; we'll use the standard terminology of "get" and "put":



Let's apply the unit-homomorphism property (6.44)



It says that put(get(s)) = s. This is typically called the get-put law.

We leave the comultiplication-homomorphism law to Exercise 6.72, 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 6.72 (Solution here). Complete Example 6.71.

1. Write out the comultiplication law from (6.44) 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 6.73 (Very well-behaved lenses are kinda boring). We saw in Exercise 6.72 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 6.75 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:

$$F \coloneqq \{f : T \to S \mid t \in T,\ \mathrm{get}(f(t)) = t \text{ and } \mathrm{put}(f(t), t) = f(t)\}.$$

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 6.74

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 $\pi\colon S \to T \times F$ given by

$$s \mapsto \big(\mathrm{get}(s), t \mapsto \mathrm{put}(s, t)\big)$$

and it is well-defined since for all $s \in S$ and $t, t' \in T$ we have $\mathrm{get}(\mathrm{put}(s, t)) = t$ by put-get and $\mathrm{put}(\mathrm{put}(s, t), t') = \mathrm{put}(s, t')$ by put-put. We also obtain a function $\pi'\colon T \times F \to S$ given by

$$(t, f) \mapsto f(t).$$

The two functions π, π' are mutually inverse: the roundtrip on S is identity because $\mathrm{put}(s, \mathrm{get}(s)) = s$ by get-put; the roundtrip on $T \times F$ is identity because $\mathrm{get}(f(t)) = t$ and $\mathrm{put}(f(t), t) = f(t)$ by assumption on $f \in F$.

Exercise 6.74 (Solution here). Let S, T, F be sets and suppose given an isomorphism $\alpha\colon S \to T \times F$.

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 6.75 (Solution here).

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 6.76 (Solution here). 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.

6.6 Products in $\mathbf{Cat}^\sharp$

Products in $\mathbf{Cat}^\sharp$ are fascinating. Given categories $\mathcal{C}$ and $\mathcal{D}$, the set of objects in their **Cat**-product is given by the product of their sets of objects, but this is not the case in $\mathbf{Cat}^\sharp$. So what is an object in $\mathcal{C} \times^\sharp \mathcal{D}$ (the usual categorical product, but taken in $\mathbf{Cat}^\sharp$)?

An object in $\mathcal{C} \times^\sharp \mathcal{D}$ is, roughly speaking, a tree for which each node is a pair of objects: some $i \in \mathcal{C}$ and some $j \in \mathcal{D}$. The edges leading out of node (i, j) are then all the morphisms emanting from i and all the morphisms emanating from j. The tree must respect identities, codomains, and composites in $\mathcal{C}$ and $\mathcal{D}$, as we'll explain. But before we do, we define the following category, which will be crucial to our understanding of products in $\mathbf{Cat}^\sharp$.

Definition 6.77 (Free monoidal categories on monoids and comonoids)**.** Given a (small) set I, define Δ_I to be the free monoidal category generated by $|I|$ distinct monoids. Dually, Δ_I^{op} is the free monoidal category generated by $|I|$ distinct comonoids.

Essentially, what this definition says is that Δ_I^{op} is a monoidal category with $|I|$ distinct comonoids, each with its own counit and comultiplication, as well as all the objects and morphisms that can be obtained from these comonoids via composition and taking the monoidal product. These objects and morphisms are then subject to no relations beyond those implied by the standard comonoid axioms.

In particular, we can identify the objects of Δ_I^{op} with the elements of the free monoid $\mathrm{List}(I)$, where the $|I|$ comonoids are the singleton lists $[i]$ for each $i \in I$. The monoidal product of Δ_I^{op} can then be interpreted as list concatenation, which—in a suggestive overloading of notation—we will denote by $\triangleleft$. The monoidal unit must then be the empty list $[]$. We could give analogous notation for Δ_I.

The counits $[i] \to []$ and comultiplications $[i] \to [i, i]$ for each $i \in I$ generate all the morphisms of Δ_I^{op} via composition and taking the monoidal product, while satisfying associativity and left and right unit laws. For instance, with $I \coloneqq \mathtt{3}$, there are two distinct morphisms $[2, 3, 1, 3] \to [2, 2, 3]$ in Δ_I^{op}. One of them is given by

$$([2] \to [2, 2]) \triangleleft \mathrm{id}_{[3]} \triangleleft ([1] \to []) \triangleleft ([3] \to []),$$

and the other is given by

$$([2] \to [2, 2]) \triangleleft ([3] \to []) \triangleleft ([1] \to []) \triangleleft \mathrm{id}_{[3]}.$$

Another way to state Definition 6.77 would be to say that Δ_I^{op} is initial among monoidal categories equipped with $|I|$ comonoids. That is, given any monoidal category $(\mathcal{C}, y, \triangleleft)$ with a comonoid $(c_i, \epsilon_i, \delta_i)$ for each $i \in I$, there is a unique monoidal functor $\Delta_I^{\mathrm{op}} \to \mathcal{C}$ that sends each $[i]$ to c_i, each $[i] \to []$ to ϵ_i, and each $[i] \to [i, i]$ to δ_i. Dually, Δ_I is initial among monoidal categories equipped with $|I|$ monoids.

Example 6.78 (The augmented simplex category). If we take $I := 1$, then $\Delta := \Delta_I$ is what is commonly known as the *augmented simplex category* or the *algebraist's simplex category*. It is the free monoidal category generated by the monoid $[1]$ with unit $[] \to [1]$ and multiplication $[1, 1] \to [1]$).

If we identify each n-element list $[1, \ldots, 1]$ with the finite set n, interpreted as an ordinal, we can see that Δ is in fact the category of all finite ordinals (i.e. $\mathsf{0}, \mathsf{1}, \mathsf{2}, \ldots$) and the order-preserving maps between them. In [**maclane**], Mac Lane verifies that Δ is initial among monoidal categories equipped with a monoid.

Armed with the free monoidal category Δ_I^{op}, we are now ready to state how products can be constructed in $\mathbf{Cat}^\sharp$.

Proposition 6.79. The category $\mathbf{Cat}^\sharp$ has all small products.

In particular, for a small set I and a category $\mathcal{C}_i \in \mathbf{Cat}^\sharp$ corresponding to a comonoid $\mathfrak{c}_i \in \mathbf{Poly}$ for each $i \in I$, the product of the $\mathcal{C}_i$'s is given by the limit of the canonical monoidal functor $C \colon \Delta_I^{\mathrm{op}} \to \mathbf{Poly}$ sending each $[i]$ to $\mathfrak{c}_i$.

Before we give the proof of the proposition above, let us examine what it says concretely in the case of binary products.

Example 6.80 (Binary products in $\mathbf{Cat}^\sharp$). For each $i \in \mathsf{2}$, let $\mathcal{C}_i$ be the category corresponding to the comonoid $(\mathfrak{c}_i, \epsilon_i, \delta_i)$ in $\mathbf{Poly}$. Then we can define $C \colon \Delta_{\mathsf{2}}^{\mathrm{op}} \to \mathbf{Poly}$ to be the canonical monoidal functor sending each $[i]$ to $\mathfrak{c}_i$. Proposition 6.79 asserts that $\lim C$ is the product of $\mathcal{C}_1$ and $\mathcal{C}_2$ in $\mathbf{Cat}^\sharp$. But what kind of a polynomial is $\lim C$, and what is its comonoid structure?

Well, for every list $\ell \in \mathrm{Ob}(\Delta_{\mathsf{2}}^{\mathrm{op}}) = \mathrm{List}(\mathsf{2})$, the polynomial $\lim C$ should have a $C\ell$-component, i.e. a projection $\pi_\ell \colon \lim C \to C\ell$. For example, when $\ell := [1, 2, 2, 1, 1, 1]$, we have $C\ell = \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1$, giving us a projection $\lim C \to \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1$. These projections must commute with the morphisms in the image of C, which are precisely the morphisms generated by the ϵ_i's and the δ_i's via composition and taking the monoidal product. For instance, the diagram

$$\begin{tikzcd}
\lim C \arrow[r, "\pi_{[1,2,1]}"] \arrow[dr, "\pi_{[1,2,2,1,1,1]}"'] & \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \arrow[d, "\mathfrak{c}_1 \triangleleft \delta_2 \triangleleft (\delta_1 \fatsemi (\mathfrak{c}_1 \triangleleft \delta_1))"] \\
& \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1 \triangleleft \mathfrak{c}_1
\end{tikzcd} \tag{6.81}$$

commutes. In fact, notice that for all $\ell \in \mathrm{List}(\mathsf{2})$, there exists a unique alternating list ℓ' of 1's and 2's with no repetitions (such as $[1, 2, 1], [2, 1, 2, 1], []$, or $[2]$) for which there is a unique morphism $d_{\ell,\ell'} \colon \ell' \to \ell$ generated by comultiplications $[i] \to [i, i]$ (here uniqueness is guaranteed by associativity). So we can generalize (6.81) to say that

$$\begin{tikzcd}
\lim C \arrow[r, "\pi_{\ell'}"] \arrow[dr, "\pi_{\ell}"'] & C\ell' \arrow[d, "Cd_{\ell,\ell'}"] \\
& C\ell
\end{tikzcd} \tag{6.82}$$

commutes.

Hence the family of projections π_ℓ for all $\ell \in \mathsf{List}(2)$ is completely characterized by just the projections $\pi_{\ell'}$ for which ℓ' contains no repetitions. Let us focus, then, on only those projections. Together, they form the commutative diagram

$$\begin{tikzcd}
y & \mathfrak{c}_1 \arrow[l, "\epsilon_1"'] & \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \arrow[l, "\mathfrak{c}_1 \triangleleft \epsilon_2"'] & \cdots \arrow[l] \\
\lim C \arrow[u, "\pi_{[]}"] \arrow[ur, "\pi_{[1]}"] \arrow[urr, "\pi_{[1,2]}"'] \arrow[d, "\pi_{[]}"'] \arrow[dr, "\pi_{[2]}"'] \arrow[drr, "\pi_{[2,1]}"] \\
y & \mathfrak{c}_2 \arrow[l, "\epsilon_2"] & \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \arrow[l, "\mathfrak{c}_2 \triangleleft \epsilon_1"] & \cdots \arrow[l] \, .
\end{tikzcd} \tag{6.83}$$

It follows that there are projections from $\lim C$ to both the limit of the top row of (6.83) and the limit of the bottom row of (6.83). In particular, the limit of the top row of (6.83) can be thought of intuitively as the "infinite" monoidal product $\mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \triangleleft \mathfrak{c}_1 \triangleleft \cdots$, corresponding to the polynomial whose positions are all of the infinite structures that can be constructed by following these instructions:

1.1. choose an object $c_1 \in \mathcal{C}_1$;
1.2. for each morphism f_1 in $\mathcal{C}_1$ with domain c_1:
    2.1. choose an object $d_2 \in \mathcal{C}_2$;
    2.2. for each morphism g_2 in $\mathcal{C}_2$ with domain d_2:
        3.1. choose an object $c_3 \in \mathcal{C}_1$;
        3.2. for each morphism f_3 in $\mathcal{C}_1$ with domain c_3:
            $\cdots$,

Then the directions at each such position are the sequences of morphisms formed by starting from the top of the instructions above and taking the morphisms mentioned in steps $1.2, 2.2, \ldots, n.2$, for some finite n, to obtain sequences such as $()$, (f_1), (f_1, g_2), and (f_1, g_2, f_3). The limit of the bottom row of (6.83) can be characterized in the same way, but with the roles of $\mathcal{C}_1$ and $\mathcal{C}_2$ swapped. From these structures, we can read off the behavior of each projection $\pi_{\ell'}$ from $\lim C$; for instance, the projection $\pi_{[1,2]}$ sends each position of $\lim C$ to the position of $\mathfrak{c}_1 \triangleleft \mathfrak{c}_2$ specified by following just the first three steps of the above instructions.

So every position of $\lim C$ yields a pair of these structures. We'll call the structure obtained by following the instructions above the *left* structure, and the structure obtained by following the instructions above, but with $\mathcal{C}_1$ and $\mathcal{C}_2$ swapped, the *right* structure. But not every such pair of structures corresponds to a position of $\lim C$; they must satisfy additional conditions, given by morphisms in the image of C that are not depicted in (6.83). For instance, the fact that the diagram

$$\begin{tikzcd}
\lim C \arrow[r, "\pi_{[1,2]}"] \arrow[dr, swap, "\pi_{[2]}"] & \mathfrak{c}_1 \triangleleft \mathfrak{c}_2 \arrow[d, "\epsilon_{1 \triangleleft \mathfrak{c}_2}"] \\
& \mathfrak{c}_2
\end{tikzcd} \tag{6.84}$$

commutes implies that, when following the above instructions, the object d_2 chosen for the identity morphism on c_1 in the left structure must also be the first object chosen when constructing the right structure. Similar diagrams render all such objects chosen for identity morphisms when constructing these structures redundant. So, in the end, we have a one-to-one correspondence between positions of $\lim C$ and pairs of structures that can be constructed by following these instructions:

1.1. choose an object $c_1 \in \mathcal{C}_1$;
1.2. for each nonidentity morphism f_1 in $\mathcal{C}_1$ with domain c_1:
    2.1. choose an object $d_2 \in \mathcal{C}_2$;
    2.2. for each nonidentity morphism g_2 in $\mathcal{C}_2$ with domain d_2:
        3.1. choose an object $c_3 \in \mathcal{C}_1$;
        3.2. for each nonidentity morphism f_3 in $\mathcal{C}_1$ with domain c_3:
            ...

for the left structure, and

1.1. choose an object $d_1 \in \mathcal{C}_2$;
1.2. for each nonidentity morphism g_1 in $\mathcal{C}_2$ with domain d_1:
    2.1. choose an object $c_2 \in \mathcal{C}_1$;
    2.2. for each nonidentity morphism f_2 in $\mathcal{C}_1$ with domain c_2:
        3.1. choose an object $d_3 \in \mathcal{C}_2$;
        3.2. for each nonidentity morphism g_3 in $\mathcal{C}_2$ with domain d_3:
            ...

for the right structure. These pairs are then the objects of $\mathcal{C}_1 \times^\sharp \mathcal{C}_2$. The morphisms from each object are the sequences of morphisms formed by starting from the top of either set of instructions above and taking the morphisms mentioned in steps 1.2, 2.2, ..., n.2, for some finite n, to obtain sequences such as $()$, (f_1), (g_1), (f_1, g_2), (g_1, f_2), (f_1, g_2, f_3) and (g_1, f_2, g_3). In particular, the identity morphism is $()$.

We illustrate how to determine the codomain of each nonidentity morphism using an example. To find the codomain of the morphism (g_1, f_2), obtained from steps 1.2 and 2.2 in the right structure, we need to determine the left and right structures of the codomain. Its right structure is easy to describe: it is simply the structure obtained by following the remaining instructions, starting from step 3.1, nested under step 2.2 for f_2; the recursive nature of these instructions ensures that this is, in fact, a valid right structure. As for the left structure of the codomain, we can construct it as follows:

- 1.1. choose the object $\mathrm{cod}(f_2) \in \mathcal{C}_1$;
- 1.2. for each nonidentity morphism f_2' in $\mathcal{C}_1$ with domain $\mathrm{cod}(f_2)$:
  - 2.1. if $f_2 \mathbin{⨾} f_2'$ is not the identity, then follow the steps nested under step 2.2 for $f_2 \mathbin{⨾} f_2'$ in the instructions above for constructing the original right structure;
  - 2.2. otherwise,

$\mathrm{cod}(f_2)$ in $\mathcal{C}_1$, then, for each nonidentity morphism f_2' in $\mathcal{C}_1$

Example 6.85 (Products of discrete categories are products of sets). Given sets S and T, consider the corresponding discrete categories Sy and Ty. Then by Example 6.80, the product $Sy \times^\sharp Ty$ is the discrete category $(S \times T)y$.

Example 6.86 (Products of one-object categories are coproducts of monoids). Given monoids M and N, consider the corresponding one-object categories y^M and y^N. Then by Example 6.80, the product $y^M \times^\sharp y^N$ is the one-object category y^{M*N}, where $M * N$ is the free product (i.e. the coproduct) of the monoids M and N.

Example 6.87 (Unexpectedly-many objects). Let $\mathcal{C} := \boxed{\overset{A}{\circ} \xrightarrow{f} \overset{B}{\circ}}$ be the walking arrow category. By Example 6.80, the product $\mathcal{C} \times^\sharp \mathcal{C}$ has infinitely many objects. Namely, it has an object for every pair of sequences (s_1, s_2), finite or infinite, that one can write in the alphabet $\{A_1, A_2, B_1, B_2\}$ subject to the following conditions:

- the sequence s_1 starts with either A_1 or B_1;
- the sequence s_2 starts with either A_2 or B_2; and
- in either sequence:
  - after A_1 comes either A_2 or B_2;
  - after A_2 comes either B_1 or A_1; and
  - after B_1 or B_2, the sequence stops.

For example, here is an object in $\boxed{\overset{A}{\circ} \xrightarrow{f} \overset{B}{\circ}} \times \boxed{\overset{A}{\circ} \xrightarrow{f} \overset{B}{\circ}}$:

$$((A_1, A_2, A_1, A_2, \ldots), (A_2, A_1, A_2, A_1, B_2))$$

Example 6.88. Let I be a set and Iy the associated discrete category, and let $(M, e, *)$ be a monoid and y^M the associated one-object category. Then by Example 6.80, the product

$Iy \times^\sharp y^M$ is the category $I^M y^M$. Its objects are functions $i\colon M \to I$, while its morphisms have the form $i \xrightarrow{m} (m' \mapsto i(m * m'))$ for each $i\colon M \to I$ and $m \in M$.

Given $i\colon M \to I$, its identity morphism is the morphism $i \xrightarrow{e} i$ corresponding to $e \in M$; given $m, n \in M$, the composite of the morphisms $i \xrightarrow{m} (m' \mapsto i(m * m')) \xrightarrow{n} (m' \mapsto i(m * n * m'))$ is the morphism $i \xrightarrow{m*n} (m' \mapsto i(m * n * m'))$.

Proof of Proposition 6.79. We first show that $\lim C$ actually is a comonoid in **Poly** by giving its counit and comultiplication. The counit $\epsilon\colon \lim C \to y$ is given by the projection $\pi_{[]}\colon \lim C \to C[]$, since $C[] \cong y$.

To construct the comultiplication, we observe that Δ_I^{op} is connected (every object has map to [] given by a monoidal product of counits), so by Theorem 5.44, $\triangleleft$ preserves Δ_I^{op}-shaped limits. As C is monoidal, it follows that

$$(\lim C) \triangleleft (\lim C) \cong \lim_{\ell \in \Delta_I^{\mathrm{op}}} \lim_{\ell' \in \Delta_I^{\mathrm{op}}} C(\ell \triangleleft \ell').$$

So specifying a map to $(\lim C) \triangleleft (\lim C)$ amounts to specifying a map to $C(\ell \triangleleft \ell')$ for every $\ell, \ell' \in \Delta_I^{\mathrm{op}}$ such that the appropriate diagrams commute. In particular, we can construct the comultiplication $\delta\colon \lim C \to (\lim C) \triangleleft (\lim C)$ by specifying the projection $\pi_{\ell \triangleleft \ell'}\colon \lim C \to C(\ell \triangleleft \ell')$ for each $\ell, \ell' \in \Delta_I^{\mathrm{op}}$. It is routine to verify that the appropriate diagrams commute.

(Verify comonoid laws)

(Give projections; check that these are comonoid morphisms)

(Verify universal property of product) □

6.7 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 6.89. 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 6.90.

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 **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 $f(c) \coloneqq f_1(c)$ or by $f(c) \coloneqq f_2(c)$ 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 6.90 (Solution here).

1. Show that 0 is a comonoid.
2. Show that 0 is initial as a comonoid. ◊

Exercise 6.91 (Solution here). If $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ is a category, show there is an induced category structure on the polynomial $2\mathfrak{c}$. ◊

Exercise 6.92 (Solution here). Check that the terminal comonoid is y. ◊

Proposition 6.93 (Porst)**.** The forgetful functor **Comon**(**Poly**) $\to$ **Poly** is comonadic.

Proof. The fact that a forgetful functor **Comon**(**Poly**) $\to$ **Poly** is comonadic if it has a right adjoint follows from Beck's monadicity theorem via a straightforward generalization of an argument given by Paré in [Par69, pp. 138-9], as pointed out by Porst in [Por19, Fact 3.1]. □

Corollary 6.94. The category $\mathbf{Cat}^\sharp$ = **Comon**(**Poly**) has all small colimits. They are created by the underlying polynomial functor **Comon**(**Poly**) $\to$ **Poly**.

Proof. It is well known that a comonadic functor creates all colimits that exist in its codomain [nLa18]. By Theorem 4.36, the category **Poly** has all small colimits. □

Proposition 6.95. Let $\mathbf{Comon}(\mathbf{Poly})_{\text{rep}}$ 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

$$\mathbf{Comon}(\mathbf{Poly})_{\text{rep}} \cong \mathbf{Mon}^{\text{op}}$$

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 $f\colon y^M \to y^N$ involves a single function $f^\sharp\colon N \to M$ that must satisfy a law coming from unitality and one coming from composition, as in Definition 6.42. The result can now be checked by hand, or seen formally as follows. Each object in the two diagrams (6.43) is representable by Exercise 5.7. The Yoneda embedding $\mathbf{Set}^{\text{op}} \to \mathbf{Poly}$ is fully faithful, so these two diagrams are equivalent to the unit and composition diagrams for monoid homomorphisms. □

Exercise 6.96 (Solution here). Let $\mathbf{Comon}(\mathbf{Poly})_{\text{lin}}$ be the full subcategory of comonoids (c, ϵ, δ) in $\mathbf{Poly}$ for which the carrier $c = My$ is linear. Show that there is an isomorphism of categories

$$\mathbf{Comon}(\mathbf{Poly})_{\text{lin}} \cong \mathbf{Set}.$$

◊

Proposition 6.97. The inclusion of linear comonoids into all comonoids has a left adjoint

$$\begin{tikzcd}\mathbf{Comon}(\mathbf{Poly}) \arrow[r, shift left, "(\mathfrak{c}\triangleleft 1)y"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Comon}(\mathbf{Poly})_{\text{lin}} \arrow[l, shift left, "Ay"]\end{tikzcd}$$

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 $f\colon \mathfrak{c} \to \mathfrak{d}$ of polynomials is cartesian (i.e. for each position $i \in \mathfrak{c}(1)$, the map $f_i^\sharp\colon \mathfrak{d}[f_1(i)] \to \mathfrak{c}[i]$ is an isomorphism).

Proposition 6.98. Every cofunctor $f\colon \mathcal{C} \nrightarrow \mathcal{D}$ 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 $f\colon \mathfrak{c} \to \mathfrak{d}$ 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 $g_1\colon \mathfrak{c}(1) \to \mathfrak{c}'(1)$ is identity (rather than merely isomorphism) on positions and similarly that for each $i \in \mathfrak{c}$ the map $h_i^\sharp\colon \mathfrak{c}'[i] \to \mathfrak{d}[h_1(i)]$ 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 $\operatorname{cod}(m) := \operatorname{cod}(f_i^\sharp(m))$, and given an emanating morphism $m' \in \mathfrak{c}'[\operatorname{cod}(m)]$, assign the composite $m \mathbin{⨟} m'$ in $\mathfrak{c}'$ to be $m \mathbin{⨟} m'$ in $\mathfrak{d}$. In Exercise 6.99 we will check that with these definitions, $\mathfrak{c}'$ is a category and both g and h are cofunctors. □

Exercise 6.99 (Solution here). We will complete the proof of Proposition 6.98, 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 6.100. 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 $f\colon \operatorname{Ob}(\mathcal{C}) \to \operatorname{Ob}(\mathcal{D})$. For any object $c \in \operatorname{Ob}(\mathcal{C})$, a functor gives a function, say $f_\sharp\colon \mathcal{C}[c] \to \mathcal{D}[f(c)]$ 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 6.101. 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 $F_1\colon \operatorname{Ob}(\mathcal{C}) \to \operatorname{Ob}(\mathcal{D})$; 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 $G(f) := F^\sharp_{G_1(d)}$. It has the correct codomain: $\operatorname{cod}(G(f)) = G_1(F_1(\operatorname{cod}(G(f))) = G_1(\operatorname{cod} f)$. 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 6.102 (Solution here). Let S be a set and consider the state category $\mathcal{S} := (Sy^S, \epsilon, \delta)$. Use Proposition 6.101 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 6.103 (Solution here). Consider the categories $\mathcal{C} := \boxed{\bullet \rightrightarrows \bullet}$ and $\mathcal{D} := \boxed{\bullet \to \bullet}$. 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. ◊

6.7.1 Dirichlet monoidal product on Cat$^\sharp$

The usual product of categories gives a monoidal operation on comonoids too, even though it is not a product in **Cat**$^\sharp$. The carrier polynomial of the product is the $\otimes$-product of the carrier polynomials.

Proposition 6.104. The Dirichlet monoidal product $(y, \otimes)$ on **Poly** extends to a monoidal structure $(y, \otimes)$ on **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 $\epsilon_{\mathcal{C} \otimes \mathcal{D}} \colon \mathfrak{c} \otimes \mathfrak{d} \to y$ 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 $\delta_{\mathcal{C} \otimes \mathcal{D}} \colon (\mathfrak{c} \otimes \mathfrak{d}) \to (\mathfrak{c} \otimes \mathfrak{d}) \triangleleft (\mathfrak{c} \otimes \mathfrak{d})$ 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 6.105. □

Exercise 6.105 (Solution here). We complete the proof of Proposition 6.104.

1. Show that $(\mathfrak{c} \otimes \mathfrak{d}, \epsilon_{\mathfrak{c} \otimes \mathfrak{d}}, \delta_{\mathfrak{c} \otimes \mathfrak{d}})$, as described in Proposition 6.104, forms a comonoid.

2. Check that the construction $(\mathcal{C}, \mathcal{D}) \mapsto \mathcal{C} \otimes \mathcal{D}$ is functorial in $\mathcal{C}, \mathcal{D} \in \mathbf{Cat}^{\sharp}$. ◊

6.8 Exercise solutions

Solution to Exercise 6.21.

We seek the number of admissible sections of the category $\boxed{\bullet \to \bullet}$. There are 2 choices of morphisms emanating from the object on the left, and 1 choice of morphism emanating from the object on the right, for a total of $2 \cdot 1 = 2$ admissible sections.

Solution to Exercise 6.28.

1. The category $\boxed{A \xrightarrow{f} B}$ has 2 morphisms out of A and 1 morphism out of B, so its emanation polynomial is $y^2 + y$.
2. The category $\boxed{B \xleftarrow{f} A \xrightarrow{g} C}$ has 3 morphisms out of A and 1 morphism out of each of B and C, so its emanation polynomial is $y^3 + 2y$.
3. The empty category has no objects, so its emanation polynomial is the empty sum 0.
4. The monoid $(\mathbb{N}, 0, +)$ has 1 object, and its morphisms form the set $\mathbb{N}$, so its emanation polynomial is $y^{\mathbb{N}}$.
5. The monoid $(M, e, *)$ has 1 object, and its morphisms form the set M, so its emanation polynomial is y^M.
6. The poset $(\mathbb{N}, \leq)$ has $\mathbb{N}$ as its set of objects, and there is exactly one morphism from every $n \in \mathbb{N}$ to each element of $\{n' \in \mathbb{N} \mid n \leq n'\} \cong \mathbb{N}$ (and no other morphisms from n), so the emanation polynomial of the poset is $\mathbb{N}y^{\mathbb{N}}$.
7. The poset $(\mathbb{N}, \geq)$ has $\mathbb{N}$ as its set of objects, and there is exactly one morphism from every $n \in \mathbb{N}$ to each element of $\{0, 1, \ldots, n\} \cong \mathsf{n} + \mathsf{1}$ (and no other morphisms from n), so the emanation polynomial of the poset is $\sum_{n \in \mathbb{N}} y^{\mathsf{n}+\mathsf{1}} \cong y^1 + y^2 + y^3 + \cdots$.

Solution to Exercise 6.34.

Here $(M, e, *)$ is a monoid, S is a set, $\alpha \colon M \times S \to S$ is an action, and $\mathcal{A}$ is the associated category with emanation polynomial Sy^M; in particular, for each $s \in S$ and $m \in M$, there is a morphism $s \xrightarrow{m} \alpha(m, s)$.

1. We wish to identify the identity morphism for each object $s \in \mathcal{A}$. This should be a morphism whose domain and codomain are s. By the laws of monoid actions, $\alpha(m, s)$ is guaranteed to be s if $m = e$. Hence, it only makes sense for the identity morphism of s to be the morphism $s \xrightarrow{e} s$.
2. Given a morphism $s \xrightarrow{m} \alpha(m, s)$, we wish to determine its composite with another morphism $\alpha(m, s) \xrightarrow{n} \alpha(n, \alpha(m, s))$. By the laws of monoid actions, we have that $\alpha(n, \alpha(m, s)) = \alpha(n * m, s)$, so it makes sense for the composite $s \xrightarrow{m} \alpha(m, s) \xrightarrow{n} \alpha(n, \alpha(m, s))$ to be the morphism $s \xrightarrow{n*m} \alpha(n * m, s)$.

Solution to Exercise 6.37.

The linear polynomial Ay corresponds to a category whose objects form the set A and whose only morphisms are identities: in other words, it is the discrete category on A.

Solution to Exercise 6.39.

1. If every object in $\mathcal{C}$ is linear, then the only morphisms in $\mathcal{C}$ are the identity morphisms, so $\mathcal{C}$ must be a discrete category.
2. It is not possible for an object in $\mathcal{C}$ to have degree 0, as every object must have at least an identity morphism emanating from it.
3. Some possible examples of categories with objects of degree $\mathbb{N}$ are the monoid $(\mathbb{N}, 0, +)$ (see Exercise 6.28 #4), the poset $(\mathbb{N}, \leq)$ (see Exercise 6.28 #6), and the state category on $\mathbb{N}$ (see Example 6.36).

4. There are 3 categories with just one linear and one quadratic object. They can be distinguished by the behavior of the single non-identity morphism. Either its domain and its codomain are distinct, in which case we have the walking arrow category; or its domain and its codomain are the same, in which case it can be composed with itself to obtain either itself or the identity, yielding two more categories for a total of three.
5. Yes: since categories correspond to comonoids, there are as many categories with one linear and one quadratic object as there are comonoid structures on $y^2 + y$.

Solution to Exercise 6.40.

1. Take the discrete category on n and adjoin a unique initial object A, so that the only non-identity morphisms are the morphisms from A to each of the other objects exactly once. Then this category has the emanation polynomial $y^{\mathsf{n}+1} + \mathsf{n}y$.
2. This category could be thought of as "star-shaped," with the initial object in the center with morphisms leading out to the other n objects as spokes.

Chapter 7

Cofree polynomial comonoids

7.1 Introduction: Cofree comonoids and discrete 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 $f\colon Sy^S \to p$. 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 $\mathfrak{s} := Sy^S$ 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 $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$, a seasoned category theorist might be tempted to ask the following: is there an adjunction

$$\mathbf{Poly}(U\mathscr{C}, p) \cong \mathbf{Cat}^\sharp(\mathscr{C}, \mathscr{T}_p),$$

for some functor $\mathscr{T}\colon \mathbf{Poly} \to \mathbf{Cat}^\sharp$? 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 7.1 (Cofree comonoid)**.** The forgetful functor **Comon**(**Poly**) $\to$ **Poly** has a right adjoint

$$\begin{tikzcd} \mathbf{Cat}^\sharp \arrow[r, shift left, "\mathfrak{c}"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift left, "\mathscr{T}_p"] \end{tikzcd}$$

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 **??**.

7.2 Cofree comonoids as trees

Definition 7.2 (Cofree comonoid)**.** Let $p \in \mathbf{Poly}$ be a polynomial. The comonoid $\mathscr{T}_p = (\mathsf{tree}_p, \text{root}, \text{focus})$ as in Theorem 7.1 is called the *cofree comonoid on p* or informally the category of *(possibly infinite) p-trees*.

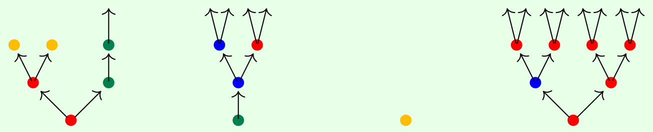
An object $t \in \mathsf{tree}_p(1)$ is a called a *(possibly infinite) tree in p*. Given such an object t in the category, an emanating morphism $n \in \mathsf{tree}_p[t]$ is called a *path from root*.

The terminology of Definition 7.2 is alluding to a specific way we like to imagine the comonoid $\mathscr{T}_p = (\mathsf{tree}_p, \text{root}, \text{focus})$, 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 7.3*.* Let $p := \{\bullet, \bullet\}y^2 + \bullet y + \bullet$. Here are four trees in p:



(7.4)

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 7.2, 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 (7.4) 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 7.5 (Solution here).

1. Interpret each of the five tree examples imagined in Example 7.3 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 $\mathscr{T}_p = (\mathsf{tree}_p, \text{root}, \text{focus})$, we just need to explain how identities, codomains, and composition work, i.e. we just need to give the counit map $\text{root}\colon \mathsf{tree}_p \to y$ and the comultiplication map $\text{focus}\colon \mathsf{tree}_p \to \mathsf{tree}_p \triangleleft \mathsf{tree}_p$.

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 7.6 (Solution here). Let $p := \{\bullet, \bullet\}y^2 + \bullet y + \bullet$ as in Example 7.3.

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 7.7. 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 7.8 (Solution here). Let A be a set.

1. What is tree_A?
2. How is it given the structure of a category? ◊

Example 7.9. 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 $t := \boxed{\bullet \to \bullet \to \bullet \to \cdots}$.

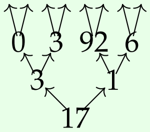
So tree_p has a single position, namely t. That position has an emanating morphism for each path out of the root, so it has $\mathbb{N}$-many emanating morphisms: one for every length. Hence $\mathsf{tree}_y = y^{\mathbb{N}}$.

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 7.10 (Solution here). Let A be a set.

1. What is tree_{Ay}?
2. How is it given the structure of a category? ◊

Example 7.11. 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 $t\colon \mathsf{List}(2) \to \mathbb{N}$, 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 $t'\colon \mathsf{List}(2) \to \mathbb{N}$ 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 7.12 (Solution here). 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 7.13 ($\mathscr{T}_{Ay}$ 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 $t\colon \mathbb{N} \to A$, 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 $n\colon t \to (t+n)$ then the composite of morphisms n, n' is $(n+n')\colon t \to (t+n+n')$.

We first saw this category in Example 6.18

Exercise 7.14 (Solution here). 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 7.15 (Solution here). 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. ◊

Exercise 7.16 (Solution here). Let p be a polynomial, let $Q \coloneqq \{q \in \mathbb{Q} \mid q \geq 0\}$ and consider the monoid y^Q of nonnegative rational numbers under addition. Is it true that any cofunctor $\varphi \colon \mathscr{T}_p \nrightarrow y^{\mathbb{Q}}$ is constant, i.e. that it factors as

$$\mathscr{T}_p \nrightarrow y \nrightarrow y^{\mathbb{Q}}?$$

◊

7.2.1 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 $\operatorname{cod}(d)$ 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 7.17 (Solution here). 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 7.18 (Solution here). 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? ◊

7.3 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.

7.3.1 Copointing

Definition 7.19 (Copointed polynomial)**.** A *copointed polynomial* is a pair (p, ϵ), where $p \in \mathbf{Poly}$ is a polynomial and $\epsilon\colon p \to y$ is a morphism in **Poly**.

A *morphism* of copointed polynomials $f\colon (p, \epsilon) \to (p', \epsilon')$ is a morphism $f\colon p \to p'$ such that $\epsilon = f \mathbin{⨟} \epsilon'$.

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 $\mathscr{T}_{-}\colon \mathbf{Poly} \to \mathbf{Comon}(\mathbf{Poly})$—and we will obtain it in two steps.

Proposition 7.20. 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

$$\begin{tikzcd} \mathbf{Cpt}(\mathbf{Poly}) \arrow[r, shift left, "py"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Poly} \arrow[l, shift left, "q"] \end{tikzcd},$$

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 $\pi\colon py \to y$, 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

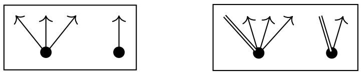
$$\begin{tikzcd} q \arrow[rr] \arrow[dr, "\epsilon"'] & & py \arrow[dl, "\pi"] \\ & y & \end{tikzcd}$$

natural in both q and p. This completes the proof. □

Exercise 7.21 (Solution here). Show that the copointing functor is essentially surjective. That is, every polynomial p equipped with a map $\epsilon\colon p \to y$ 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 $\mathscr{T}_p = (\mathsf{tree}_p, \text{root}, \text{focus})$. 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 6.15. 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 7.22 (Slowing down dynamical systems). Given a dynamical system $f\colon Sy^S \to p$, we automatically get a map $Sy^S \to py$ to the cofree pointing. We called this "adding a pause button" in Example 3.35. 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 (6.13). 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 $\epsilon\colon p \to y$, 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 $f\colon Sy^S \to p$, 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 7.23 (Solution here).

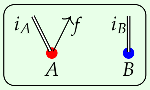
1. Show that there is a monoidal structure $(y, \otimes)$ on **Cpt**(**Poly**) such that the forgetful functor $U\colon \mathbf{Cpt}(\mathbf{Poly}) \to \mathbf{Poly}$ is strong monoidal.
2. Show that this monoidal structure is closed, i.e. that there is an internal hom $[-,-]$ on **Cpt**(**Poly**). Hint: you should have $U([py, qy]_{\mathbf{Cpt}}) \cong [py, q]_{\mathbf{Poly}}y$. ◊

7.3.2 Constructing the cofree comonoid

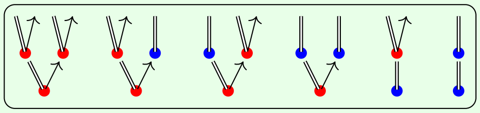
It remains to show that we can functorially take any copointed polynomial (q, ϵ) and return a comonoid, and that this construction is right adjoint to the forgetful functor. From the description in Section 7.2, we know the cofree comonoid is supposed to have something to do with infinite trees. And we 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 $\epsilon\colon q \to y$, we have two maps $q \triangleleft q \rightrightarrows q$, namely $q \triangleleft \epsilon$ and $\epsilon \triangleleft q$.

Example 7.24. Suppose $q = \{A\}y^{\{i_A, f\}} + \{B\}y^{i_B}$ with copointing ϵ selecting i_A and i_B:

$q :=$ 

Then $q \triangleleft q$ looks as follows

$q \triangleleft q =$ 

How can we picture the maps $(q \triangleleft \epsilon), (\epsilon \triangleleft q)\colon q \triangleleft q \to q$?

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

$$\begin{tikzcd}
y & q \arrow[l, "\epsilon"'] & q \triangleleft q \arrow[l, shift right, "\epsilon \triangleleft q"'] \arrow[l, shift left, "q \triangleleft \epsilon"] & q^{\triangleleft 3} \arrow[l, shift right=2, "\epsilon \triangleleft q \triangleleft q"'] \arrow[l, "q \triangleleft \epsilon \triangleleft q" description] \arrow[l, shift left=2, "q \triangleleft q \triangleleft \epsilon"] & \cdots \arrow[l, shift right=3] \arrow[l, shift right=1] \arrow[l, shift left=1] \arrow[l, shift left=3]
\end{tikzcd} \tag{7.25}$$

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 $Q\colon \Delta_+ \to \mathbf{Poly}$ as above, and this is functorial in q.

Theorem 7.26. For any copointed polynomial $q \xrightarrow{\epsilon} y$, let $\bar{q}$ denote the limit of $Q\colon \Delta_+ \to \mathbf{Poly}$. It naturally has the structure of a comonoid $(\bar{q}, \epsilon, \delta)$, and this construction is right adjoint to the forgetful functor

$$\begin{tikzcd}
\mathbf{Comon}(\mathbf{Poly}) \arrow[r, shift left=2, "U"] \arrow[r, phantom, "\Rightarrow"] & \mathbf{Cpt}(\mathbf{Poly}) \arrow[l, shift left=2, "\bar{q}"]
\end{tikzcd} .$$

Proof sketch. We first give $\bar{q}$ the structure of a comonoid. Since $\bar{q}$ is the limit of (7.25), the inclusion the inclusion $\{0\} \to \Delta_+$ induces a projection map $\bar{q} \to y$, which we again call ϵ. Since $\mathbin{\triangleleft}$ commutes with connected limits in both variables and Δ_+ is connected, we have that $\bar{q} \mathbin{\triangleleft} \bar{q}$ is the limit of the following $\Delta_+ \times \Delta_+$-shaped diagram:

$$\bar{q} \mathbin{\triangleleft} \bar{q} \cong \lim \left(\begin{tikzcd}
q^{\triangleleft(0+0)} & q^{\triangleleft(0+1)} \arrow[l] & q^{\triangleleft(0+2)} \arrow[l, shift left] \arrow[l, shift right] & q^{\triangleleft(0+3)} \arrow[l, shift left=2] \arrow[l] \arrow[l, shift right=2] & \cdots \arrow[l, shift left=3] \arrow[l, shift left=1] \arrow[l, shift right=1] \arrow[l, shift right=3] \\
q^{\triangleleft(1+0)} \arrow[u] & q^{\triangleleft(1+1)} \arrow[l] \arrow[u] & q^{\triangleleft(1+2)} \arrow[l, shift left] \arrow[l, shift right] \arrow[u] & q^{\triangleleft(1+3)} \arrow[l, shift left=2] \arrow[l] \arrow[l, shift right=2] \arrow[u] & \cdots \arrow[l, shift left=3] \arrow[l, shift left=1] \arrow[l, shift right=1] \arrow[l, shift right=3] \\
q^{\triangleleft(2+0)} \arrow[u, shift left] \arrow[u, shift right] & q^{\triangleleft(2+1)} \arrow[l] \arrow[u, shift left] \arrow[u, shift right] & q^{\triangleleft(2+2)} \arrow[l, shift left] \arrow[l, shift right] \arrow[u, shift left] \arrow[u, shift right] & q^{\triangleleft(2+3)} \arrow[l, shift left=2] \arrow[l] \arrow[l, shift right=2] \arrow[u, shift left] \arrow[u, shift right] & \cdots \arrow[l, shift left=3] \arrow[l, shift left=1] \arrow[l, shift right=1] \arrow[l, shift right=3] \\
q^{\triangleleft(3+0)} \arrow[u, shift left=2] \arrow[u] \arrow[u, shift right=2] & q^{\triangleleft(3+1)} \arrow[l] \arrow[u, shift left=2] \arrow[u] \arrow[u, shift right=2] & q^{\triangleleft(3+2)} \arrow[l, shift left] \arrow[l, shift right] \arrow[u, shift left=2] \arrow[u] \arrow[u, shift right=2] & q^{\triangleleft(3+3)} \arrow[l, shift left=2] \arrow[l] \arrow[l, shift right=2] \arrow[u, shift left=2] \arrow[u] \arrow[u, shift right=2] & \cdots \arrow[l, shift left=3] \arrow[l, shift left=1] \arrow[l, shift right=1] \arrow[l, shift right=3] \\
\vdots \arrow[u, shift left=3] \arrow[u, shift left=1] \arrow[u, shift right=1] \arrow[u, shift right=3] & \vdots \arrow[u, shift left=3] \arrow[u, shift left=1] \arrow[u, shift right=1] \arrow[u, shift right=3] & \vdots \arrow[u, shift left=3] \arrow[u, shift left=1] \arrow[u, shift right=1] \arrow[u, shift right=3] & \vdots \arrow[u, shift left=3] \arrow[u, shift left=1] \arrow[u, shift right=1] \arrow[u, shift right=3] & \ddots
\end{tikzcd} \right)$$

There is a commutative diagram in **Cat**

$$\begin{tikzcd}
\Delta_+ \times \Delta_+ \arrow[rr, "+"] \arrow[dr, bend right, "(m_1{,}m_2)\mapsto q^{\triangleleft(m_1+m_2)}"'] & & \Delta_+ \arrow[dl, bend left, "n\mapsto q^{\triangleleft n}"] \\
& \mathbf{Poly} &
\end{tikzcd} \tag{7.27}$$

which induces a map (in the opposite direction) between their limits $\delta\colon \bar{q} \to \bar{q} \mathbin{\triangleleft} \bar{q}$, which we take to be the comultiplication. Appending (7.27) 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 (7.25). 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 6.4. 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

$$\begin{tikzcd}
\mathfrak{c} \arrow[r, "\delta^{m+n-1}"] \arrow[d, "\delta"'] & \mathfrak{c}^{\triangleleft(m+n)} \arrow[d, equal] \\
\mathfrak{c} \triangleleft \mathfrak{c} \arrow[r, "\delta^{m-1} \triangleleft \delta^{n-1}"'] & \mathfrak{c}^{\triangleleft m} \triangleleft \mathfrak{c}^{\triangleleft n}
\end{tikzcd}$$

commutes for any $m, n \in \mathbb{N}$. Both triangle equations are straightforward. □

Remark 7.28. The construction of the cofree comonoid from a copointed endofunctor in the proof of Theorem 7.26 is fairly standard; see [Lac10]. Nelson Niu has also constructed the cofree comonoid on a polynomial using a different limit diagram

$$\begin{tikzcd}
y \arrow[d] & p \arrow[d] & p \triangleleft p \arrow[d] & p \triangleleft p \triangleleft p \arrow[d] & \cdots \\
1 & p \triangleleft 1 \arrow[l, "!"'] & p \triangleleft p \triangleleft 1 \arrow[l, "p \triangleleft !"'] & p \triangleleft p \triangleleft p \triangleleft 1 \arrow[l, "p \triangleleft p \triangleleft !"'] & \cdots \arrow[l]
\end{tikzcd}$$

in terms of the original polynomial p, rather than from its copointing py; that is, this construction is right adjoint to **Comon**(**Poly**) $\to$ **Poly**. One could also construct this right adjoint using the following limit, again applied to the original polynomial p:

$$\begin{tikzcd}
y \arrow[dr] & & p \arrow[dl] \arrow[dr] & & p \triangleleft p \arrow[dl] \arrow[dr] & & p \triangleleft p \triangleleft p \arrow[dl] \arrow[dr] & \cdots \\
& 1 & & p \triangleleft 1 & & p \triangleleft p \triangleleft 1 & & \cdots
\end{tikzcd}$$

We record the following proposition here; it will be useful in Corollary 10.7.

Proposition 7.29. If $f\colon p \to q$ is a Cartesian map of polynomials, then $\mathsf{tree}_f\colon \mathsf{tree}_p \to \mathsf{tree}_q$ is a Cartesian cofunctor. That is, for each tree $t \in \mathsf{tree}_p(1)$ the function $\mathsf{tree}_f^\sharp\colon \mathsf{tree}_p[t] \xrightarrow{\cong} \mathsf{tree}_q[\mathsf{tree}_f(t)]$ is a bijection.

Proof. ** □

Proposition 7.30. 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 $p, q \in$ **Poly**.

7.4 $\mathbf{Sys}(p)$ is a topos

Theorem 7.31. Let $\mathscr{T}_p$ be the cofree comonoid on $p \in \mathbf{Poly}$. There is an equivalence of categories

$$\mathbf{Sys}(p) \cong \mathbf{Cat}(\mathscr{T}_p, \mathbf{Set})$$

between the category of dynamical systems on p and that of functors $\mathscr{T}_p \to \mathbf{Set}$.

Proof. ** (In general, coalgebras of comonoids are copresheaves on the corresponding category) □

A consequence of this is that $\mathbf{Sys}(p)$ forms a topos, and hence has a ready-made type theory and internal logic. While we don't have space to do this justice, we will briefly discuss the sort of logical statement one can make about dynamical systems.

7.5 Morphisms between cofree comonoids

Given a morphism of polynomials $\varphi \colon p \to q$, the cofree functor gives us a map of comonoids $\mathscr{T}_\varphi \colon \mathscr{T}_p \to \mathscr{T}_q$, which works as follows.

An object $t \in \mathsf{tree}_p$ is a tree; the tree $u := \mathscr{T}_\varphi(t) \in \mathsf{tree}_q$ is constructed recursively as follows. If the root of t is $i \in p(1)$ then the root of u is $j := \varphi_1(i)$. To each branch $e \in q[j]$, we need to assign a new tree, and we use the one situated at $\varphi_i^\sharp(e)$.

Exercise 7.32 (Solution here). Let $p := y^2 + 1$ and $q := 2y + 2$.

1. Choose a map $\varphi \colon p \to q$, and write it out.
2. Choose a tree $t \in \mathsf{tree}_p$ with at least height 2.
3. What is $\mathscr{T}_\varphi(t)$? ◊

Exercise 7.33 (Solution here). The following exercise is useful when considering the (topos-theoretic) logic of dynamical systems. Namely, it will allow us to specify legal subtrees of height n.

1. Choose a polynomial q and a map $\epsilon \colon q \to y$, i.e. a copointed polynomial.
2. Recall from Theorem 7.26 that the carrier $\bar{q}$ of the cofree comonoid $\mathscr{T}_q = (\bar{q}, \epsilon, \delta)$ is constructed as a limit

$$\bar{q} = \lim\Bigg(\begin{tikzcd} y & q \arrow[l] & q \triangleleft q \arrow[l, shift left] \arrow[l, shift right] & q^{\triangleleft 3} \arrow[l] \arrow[l, shift left=2] \arrow[l, shift right=2] & \cdots \arrow[l, shift left=1] \arrow[l, shift right=1] \arrow[l, shift left=3] \arrow[l, shift right=3] \end{tikzcd} \Bigg)$$

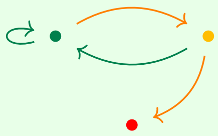
   and in particular there is a structure map $\bar{q} \to q^{\triangleleft n}$, for any $n \in \mathbb{N}$. Where does it send a tree $t \in \mathsf{tree}_q$?
3. There is an induced cofunctor $\mathscr{T}_q \to \mathscr{T}_{q^{\triangleleft n}}$. Show that for all $n \geq 1$, it is it an isomorphism. ◊

7.6 Consequences of adjointness

Recall from Definition 3.30 that a dependent system (or generalized Moore machine) is a map of polynomials $f\colon Sy^S \to p$. 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 7.34. Let $S := \{\bullet, \bullet, \bullet\}$ and $p := y^2 + y$, and consider the dynamical system $f\colon Sy^S \to p$ from Exercise 3.32, depicted here again for your convenience:

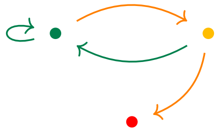


The polynomial map f is supposed to induce a cofunctor $F\colon Sy^S \nrightarrow \mathscr{T}_p$ from the state category on S to the category of p-trees. Thus to each state $s \in S$, we need to associate a tree; which should it be?

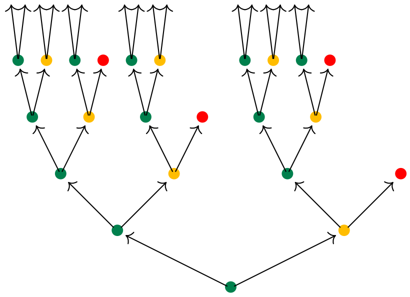
Exercise 7.35 (Solution here). Consider the walking arrow category $\mathcal{W} = \boxed{\bullet \to \bullet}$. Draw the cofunctor $\mathcal{W} \nrightarrow \mathscr{T}_{y^2+y}$. $\Diamond$

7.6.1 Dynamical systems and graph fibrations

In Example 7.34, 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 7.36 (Solution here).

1. Draw the other two trees associated to the dynamical system in Example 7.34.
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? ◊

7.6.2 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 $F\colon \mathcal{C} \to \mathcal{T}_p$, 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 $\text{cod}(F_c^\sharp(d)) \in \mathcal{C}$.

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 $\mathcal{T}_p$, 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 Part III that there are strong theoretical reasons 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 $p \in \textbf{Poly}$.

7.7 Some math about cofree comonoids

Proposition 7.37. 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 7.38. 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 7.37. □

Proposition 7.39. 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 6.104, the left adjoint $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is strong monoidal. A consequence of Kelly's doctrinal adjunction theorem [Kel74] says that the right adjoint of an op-lax monoidal functor is lax monoidal. □

Exercise 7.40 (Solution here).

1. What polynomial is tree_y?
2. What is the map $y \to \mathsf{tree}_y$ from Proposition 7.39?
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 7.39, for arbitrary $p, q \in \mathbf{Poly}$. ◊

Proposition 7.41. 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 $y^{\mathsf{List(n)}} \cong \mathscr{T}_{y^{\mathsf{n}}}$ 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 $m\colon y^{\mathsf{List(2)}} \to y^{\mathbb{N}}$ 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 $y^{\mathsf{List(2)}} \to y$ given by the list $[1,2]$. One can check by hand that these definitions make $(y^{\mathbb{N}}, e, m)$ a monoid in $(\mathbf{Cat}^{\sharp}, y, \times)$. □

Recall from Example 6.20 that an admissible section of a category $\mathcal{C}$ is a cofunctor $\mathcal{C} \nrightarrow y^{\mathbb{N}}$.

Corollary 7.42. For any category $\mathcal{C}$, the set $\mathbf{Cat}^{\sharp}(\mathcal{C}, y^{\mathbb{N}})$ of admissible sections has the structure of a monoid. Moreover, this construction is functorial

$$\mathbf{Cat}^{\sharp}(-, y^{\mathbb{N}})\colon \mathbf{Cat}^{\sharp} \to \mathbf{Mon}^{\mathrm{op}}$$

Proof. We saw that $y^{\mathbb{N}}$ is a monoid object in Proposition 7.41. □

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.

We use the notation $\mathcal{C} \mapsto \vec{\mathcal{C}}$ for the monoid of admissible sections.

Theorem 7.43. The admissible sections functor

$$\mathbf{Cat}^{\sharp} \to \mathbf{Mon}^{\mathrm{op}}$$

is right adjoint to the inclusion $\mathbf{Mon}^{\mathrm{op}} \to \mathbf{Cat}^{\sharp}$ from Proposition 6.50.

Proof. 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 $c' := \mathrm{cod}(F_c^{\sharp}(m))$. This assignment must send identities to identities and composites to composites: given $m' \in M$ we have $F_c^{\sharp}(m \mathbin{⨟} m') = F_c^{\sharp}(m) \mathbin{⨟} F_{c'}^{\sharp}(m')$. This is exactly the data of a monoid morphism $M \to \vec{\mathcal{C}}$: it assigns to each $m \in M$ an admissible section $\mathcal{C}$, preserving unit and multiplication. □

Proposition 7.44. There is a commutative square of left adjoints

$$\begin{tikzcd}
\mathbf{Mon}^{\mathrm{op}} \arrow[r, "U"] \arrow[d, "y^-"'] & \mathbf{Set}^{\mathrm{op}} \arrow[d, "y^-"] \\
\mathbf{Cat}^\sharp \arrow[r, "U"'] & \mathbf{Poly}
\end{tikzcd}$$

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 6.50, 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 4.12, that $U\colon \mathbf{Cat}^\sharp \to \mathbf{Poly}$ is a left adjoint in Theorem 7.1, and that $y^-\colon \mathbf{Mon} \to \mathbf{Cat}^\sharp$ is a left adjoint in Theorem 7.43. □

7.8 Exercise solutions

Solution to Exercise 7.16.

Take $p := 2y$, and consider the object $x \in \mathscr{T}_p(1)$ given by the stream

$$x := (2\,12\,112\,1112\,11112\,111112\ldots)$$

(with spaces only for readability); note that every morphism emanating from x has a different codomain. We need to give $\varphi_i^\sharp(q)$ for every $i \in \mathscr{T}_p(1)$ and $q \geq 0$. Define

$$\varphi_i^\sharp(q) := \begin{cases} i & \text{if } i \neq x \text{ or } q = 0 \\ x' & \text{if } i = x \text{ and } q > 0 \end{cases}$$

where $x' := (12\,112\,1112\,11112\,111112\ldots)$. There are three cofunctor conditions to check, namely identity, codomains, and composition. The codomain condition is vacuous since $y^{\mathbb{Q}}$ has one object, and the identity condition holds by construction, because we always have $\varphi_i^\sharp(0) = i$. Now take $q_1, q_2 \in Q$; we need to check that

$$\varphi^\sharp_{\operatorname{cod}\varphi_i^\sharp(q_1)}(q_2) \stackrel{?}{=} \varphi_i^\sharp(q_1 + q_2)$$

holds. If $i \neq x$ or $q_1 = q_2 = 0$, then it holds because both sides equal i. If $i = x$ and either $q_1 > 0$ or $q_2 > 0$, it is easy to check that both sides equal x', so again it holds.

Part III

Data dynamics

Chapter 8

Copresheaves, databases, and dynamical systems

In the previous part 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 part 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-co-modules, 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 $I\colon \mathcal{C} \to \mathbf{Set}$. Here's an example of a theorem we'll prove:

Theorem 8.1. For a comonoid $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$, also understood as a 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. 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 9.9.

The plan of the part is as follows. We'll begin in Chapter 8 by reviewing copresheaves on a category, and their relationship to databases and dynamical systems. Then in Chapter 9 we'll prove a number of theoretical results, including Theorem 9.9 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 10.3 we'll provide some looser discussion and lay out some open questions.

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 $\mathcal{C} \to \mathbf{Set}$ and whose morphisms are natural transformations between them.

Example 8.2. 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, including the identity and the multiplication, are encoded as the morphisms $\blacktriangle \to \blacktriangle$ in $\mathcal{G}$.

It turns out that G-sets are precisely functors $F \colon \mathcal{G} \to \mathbf{Set}$: the set $F(\blacktriangle)$ is our X

---

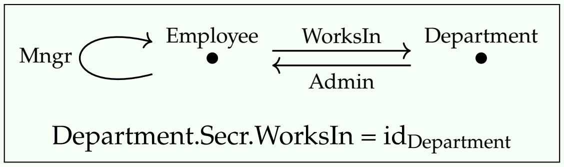
[1] Many would say that presheaves on $\mathcal{C}$ are more fundamental, but since the notions are equivalent—just use $\mathcal{C}^{\mathrm{op}}$ to switch between them—we will consider the difference moot. We will focus on copresheaves.

above, and since the elements of G are now morphisms $g\colon \blacktriangle \to \blacktriangle$, 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 8.3 (Solution here). 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 8.4 (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 $\pi(\bar{f}) = f$.

$$\begin{tikzcd}
s \arrow[r, dashed, "\bar{f}"] \arrow[d, mapsto, "\pi"'] & s' \arrow[d, mapsto, "\pi"] \\
\pi(s) \arrow[r, "f"'] & c'
\end{tikzcd}$$

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:

$$\begin{tikzcd}
\mathcal{S} \arrow[rr, "F"] \arrow[dr, "\pi"'] & & \mathcal{S}' \arrow[dl, "\pi'"] \\
& \mathcal{C} &
\end{tikzcd} \tag{8.5}$$

We denote the category of discrete opfibrations into $\mathcal{C}$ by **dopf**$(\mathcal{C})$.

Exercise 8.6 (Solution here). Show that if $F\colon \mathcal{S} \to \mathcal{S}'$ is a functor making the triangle (8.5) commute, then F is also a discrete opfibration. ◊

Exercise 8.7 (Solution here). Suppose $\pi\colon \mathcal{S} \to \mathcal{C}$ is a discrete opfibration and $i \in \mathcal{S}$ is an object. With notation as in Definition 8.4, 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 $f\colon \pi(i) \to c$ and $g\colon c \to c'$, we have $\bar{f} \mathbin{⨟} \bar{g} = \overline{f \mathbin{⨟} g}$.
3. Show that π is a cofunctor. ◊

Definition 8.8 (Category of elements $\int^{\mathcal{C}} I$)**.** Given a functor $I\colon \mathcal{C} \to \mathbf{Set}$, 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

$$\mathrm{Hom}((c, x), (c', x')) \coloneqq \{f\colon c \to c' \mid I(f)(x) = x'\}.$$

Identities and composites in $(\int^{\mathcal{C}} I)$ are inherited from $\mathcal{C}$.

There is a functor $\pi\colon (\int^{\mathcal{C}} I) \to \mathcal{C}$ sending $\pi(c, x) \coloneqq c$ and $\pi(f) \coloneqq f$.

Exercise 8.9 (Solution here). Show that if $I\colon \mathcal{C} \to \mathbf{Set}$ is a functor then the functor $\pi\colon (\int^{\mathcal{C}} I) \to \mathcal{C}$ defined in Definition 8.8 is a discrete opfibration, as in Definition 8.4. ◊

Exercise 8.10 (Solution here). Draw the category of elements for the functor $S\colon \mathcal{C} \to \mathbf{Set}$ shown in Exercise 8.3. ◊

Exercise 8.11 (Solution here). Suppose that $I, J\colon \mathcal{C} \to \mathbf{Set}$ are functors and $\alpha\colon I \to J$ 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 8.4.
3. Show that this construction is functorial. We denote this functor by

$$\int^{\mathcal{C}}\colon \mathcal{C}\text{-}\mathbf{Set} \to \mathbf{dopf}(\mathcal{C})$$

◊

Exercise 8.12 (Solution here). Let G be a graph, and let $\mathcal{G}$ be the free category on it. Show that for any functor $S\colon \mathcal{G} \to \mathbf{Set}$, the category $\int^{\mathcal{G}} S$ of elements is again free on a graph. ◊

Proposition 8.13. 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 8.11 we have a functor $\int^{\mathcal{C}}\colon \mathcal{C}\text{-}\mathbf{Set} \to \mathbf{dopf}(\mathcal{C})$. There is a functor going back: given a discrete opfibration $\pi\colon \mathcal{S} \to \mathcal{C}$, we define $(\partial\pi)\colon \mathcal{C} \to \mathbf{Set}$ on $c \in \mathrm{Ob}(\mathcal{C})$ and $f\colon c \to c'$ by

$$\begin{aligned}(\partial\pi)(c) &\coloneqq \{s \in \mathcal{S} \mid \pi(s) = c\}\\ (\partial\pi)(f)(s) &\coloneqq \bar{f}(s).\end{aligned}$$

On objects, the roundtrip $\mathcal{C}$-**Set** $\to$ $\mathcal{C}$-**Set** sends $I\colon \mathcal{C} \to \mathbf{Set}$ to the functor

$$\begin{aligned}c \mapsto &\{s \in \int^{\mathcal{C}} I \mid \pi(s)\\ &= \{(c, x) \mid x \in I(c)\} \qquad\qquad = I(c).\end{aligned}$$

The roundtrip $\mathbf{dopf}(\mathcal{C}) \to \mathbf{dopf}(\mathcal{C})$ sends $\pi\colon \mathcal{S} \to \mathcal{C}$ 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 $(\varphi_1, \varphi^\sharp)\colon \mathcal{S} \nrightarrow \mathcal{C}$ gives a function $\varphi\colon \mathrm{Ob}(\mathcal{S}) \to \mathrm{Ob}(\mathcal{C})$ and for each $s \in \mathcal{S}$ an isomorphism

$$\varphi^\sharp_s\colon \mathcal{C}[\varphi_1(s)] \xrightarrow{\cong} \mathcal{S}[s]$$

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 $\pi\colon \mathcal{S} \to \mathcal{C}$, we define $\varphi_1 := \mathrm{Ob}(\pi)$ 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 $\varphi_s^\sharp(f) := \bar{f}$ to be the lift guaranteed by Definition 8.4. 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 8.14 (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.

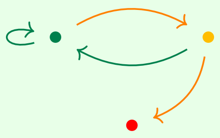
Proof. Given a discrete opfibration $\pi\colon \mathcal{S} \to \mathcal{C}$, take $S := \mathrm{Ob}(\mathcal{S})$ and define $(\varphi_1, \varphi^\sharp)\colon Sy^S \to \mathfrak{c}$ by $\varphi_1 = \pi$ and with $\varphi^\sharp$ given by the lifting: $\varphi(g) := \hat{g}$ as in Definition 8.4. One checks using Exercise 8.7 that this defines a cofunctor.

Conversely, given a cofunctor $(\varphi_1, \varphi^\sharp)\colon Sy^S \to \mathfrak{c}$, 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 8.15. □

Exercise 8.15 (Solution here). With notation as in Proposition 8.14, 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. ◊

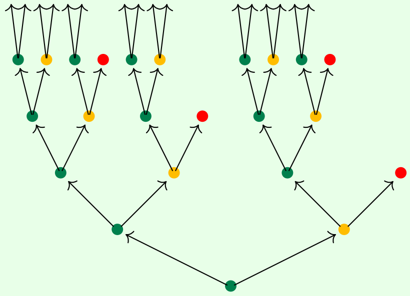
Example 8.16. In Example 7.34 we had a dynamical system with $S := \{●, ●, ●\}$ and $p := y^2 + y$, and $f\colon Sy^S \to p$ from Exercise 3.32, depicted here again for your convenience:



(8.17)

The polynomial map f induces a cofunctor $F\colon Sy^S \nrightarrow \mathcal{T}_p$ from the state category on S to the category of p-trees. We can now see this as a database instance on $\mathcal{T}_p$, considered as a database schema.

The cofree category $\mathscr{T}_t$ is actually the free category on a graph, as we saw in Proposition 7.37, 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 8.13 and 8.14, we can consider it as a discrete opfibration over $\mathscr{T}_p$. By Exercise 8.12 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 (8.17).

Exercise 8.18 (Solution here). Give an example of a dynamical system on $p := y^2 + y$ for which the corresponding database instance has a table with at least two rows. ◊

Exercise 8.19 (Solution here). 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. ◊

8.1 Exercise solutions

Chapter 9

Bimodules over polynomial comonoids

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 $\mathcal{C} \overset{m}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathcal{D}$ 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 9.1 (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 $\mathcal{C} \overset{m}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathcal{D}$ or in shorthand $\mathfrak{c} \overset{m}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$,[a] consists of

1. a polynomial $m \in$ **Poly**,
2. a morphism $\mathfrak{c} \triangleleft m \xleftarrow{\lambda} m$, and
3. a morphism $m \xrightarrow{\rho} m \triangleleft \mathfrak{d}$,

satisfying the following commutative diagrams:

$$\begin{tikzcd}
\mathfrak{c} \triangleleft m \arrow[d, "\epsilon \triangleleft m"'] & m \arrow[l, "\lambda"'] \arrow[dl, equal, bend left] \\
m &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathfrak{c} \triangleleft m \arrow[d, "\delta \triangleleft m"'] & m \arrow[l, "\lambda"'] \arrow[d, "\lambda"] \\
\mathfrak{c} \triangleleft \mathfrak{c} \triangleleft m & \mathfrak{c} \triangleleft m \arrow[l, "\mathfrak{c} \triangleleft \lambda"]
\end{tikzcd}
\tag{9.2}$$

$$\begin{tikzcd}
m \arrow[r, "\rho"] \arrow[dr, equal, bend right] & m \triangleleft \mathfrak{d} \arrow[d, "m \triangleleft \epsilon"] \\
& m
\end{tikzcd}
\qquad
\begin{tikzcd}
m \arrow[r, "\rho"] \arrow[d, "\rho"'] & m \triangleleft \mathfrak{d} \arrow[d, "m \triangleleft \delta"] \\
m \triangleleft \mathfrak{d} \arrow[r, "\rho \triangleleft \mathfrak{d}"'] & m \triangleleft \mathfrak{d} \triangleleft \mathfrak{d}
\end{tikzcd} \tag{9.3}$$

$$\begin{tikzcd}
m \arrow[r, "\rho"] \arrow[d, "\lambda"'] & m \triangleleft \mathfrak{d} \arrow[d, "\lambda \triangleleft \mathfrak{d}"] \\
\mathfrak{c} \triangleleft m \arrow[r, "\mathfrak{c} \triangleleft \rho"'] & \mathfrak{c} \triangleleft m \triangleleft \mathfrak{d}
\end{tikzcd} \tag{9.4}$$

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:

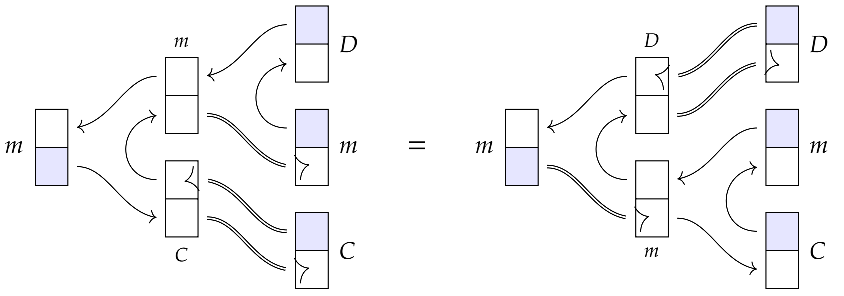
$$\begin{tikzcd}
\mathfrak{c} \triangleleft m \arrow[d] & m \arrow[l] \arrow[r] \arrow[d] & m \triangleleft \mathfrak{d} \arrow[d] \\
\mathfrak{c} \triangleleft n & n \arrow[l] \arrow[r] & n \triangleleft \mathfrak{d}
\end{tikzcd}$$

We denote the category of $(\mathcal{C}, \mathcal{D})$-bimodules by ${}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}$.

---

[a] Note that the symbol $\mathfrak{c} \triangleleft\!\!\longleftarrow\!\!\triangleleft \mathfrak{d}$ 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. (9.4) using polyboxes.

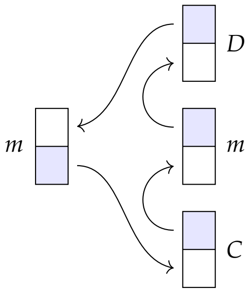


(9.5)

Exercise 9.6 (Solution here).

1. Draw the equations of Eq. (9.2) using polyboxes.
2. Draw the equations of Eq. (9.3) using polyboxes. ◊

Note that Eq. (9.5) means that we can unambiguously write



Exercise 9.7 (Solution here). 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 9.8 ($\mathscr{C}$-coalgebra)**.** Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid. A *$\mathscr{C}$-coalgebra* consists of a set S and a function $\alpha\colon S \to \mathfrak{c} \triangleleft S$, making the two diagrams below commute:

$$\begin{tikzcd}
\mathfrak{c} \triangleleft S \arrow[d, "\epsilon\triangleleft"'] & S \arrow[l, "\alpha"'] \arrow[dl, equal, bend left] \\
S &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathfrak{c} \triangleleft S \arrow[d, "\delta\triangleleft S"'] & S \arrow[l, "\alpha"'] \arrow[d, "\alpha"] \\
\mathfrak{c} \triangleleft \mathfrak{c} \triangleleft m & \mathfrak{c} \triangleleft S \arrow[l, "\mathfrak{c}\triangleleft\alpha"]
\end{tikzcd}$$

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 9.1.

In order to as quickly as possible orient the reader to bimodules, we begin by proving the following.

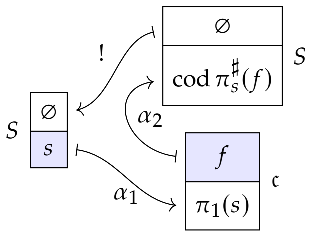
Theorem 9.9. For a comonoid $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ (category $\mathcal{C}$), the following categories are equivalent:

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 8.13.

$3 \cong 4$**:** 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



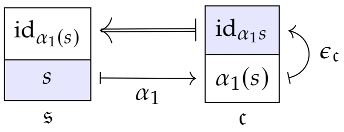
We check that this is indeed a coalgebra using properties of cofunctors. For identities in $\mathscr{C}$, we have

$$\begin{aligned}\alpha_2(s, \mathrm{id}_{\pi_1(s)}) &= \operatorname{cod} \pi^\sharp_s(\mathrm{id}_{\pi_1(s)}) \\ &= \operatorname{cod} \mathrm{id}_s = s\end{aligned}$$

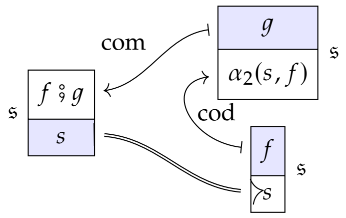
and for compositions in $\mathscr{C}$ we have

$$\begin{aligned}\alpha_2(s, f \mathbin{⨾} g) &= \operatorname{cod}\left(\pi^\sharp_s(f \mathbin{⨾} g)\right) \\ &= \operatorname{cod}\left(\pi^\sharp_s(f) \mathbin{⨾} \pi^\sharp_{\operatorname{cod} \pi^\sharp_s(f)} g\right) \\ &= \operatorname{cod}\left(\pi^\sharp_{\operatorname{cod} \pi^\sharp_s(f)} g\right) \\ &= \alpha_2(\alpha_2(s, f), g).\end{aligned}$$

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 4.64. 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 $⨾$ from $\mathfrak{c}$:



In Exercise 9.10 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.

$4 \cong 5$: This is straightforward and was mentioned in Definition 9.8.

$5 \cong 6$: A right 0-module is in particular a polynomial $m \in \mathbf{Poly}$ and a map $\rho\colon m \to m \triangleleft 0$ such that $(m \triangleleft \epsilon) \circ \rho = \mathrm{id}_m$. This implies ρ is monic, which itself implies by Proposition 4.17 that m must be constant since $m \triangleleft 0$ 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 (9.3) commute.

$5 \cong 7$: By the adjunction in Proposition 1.21 and the fully faithful inclusion $\mathbf{Set} \to \mathbf{Poly}$ of sets as constant polynmials, Proposition 4.2, we have isomorphisms

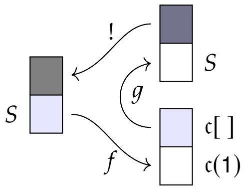
$$\mathbf{Poly}(Sy, \mathfrak{c} \triangleleft Sy) \cong \mathbf{Set}(S, \mathfrak{c} \triangleleft Sy \triangleleft 1) = \mathbf{Set}(S, \mathfrak{c} \triangleleft S) \cong \mathbf{Poly}(S, \mathfrak{c} \triangleleft S).$$

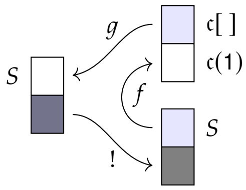
One checks easily that if $Sy \to \mathfrak{c} \triangleleft Sy)$ corresponds to $S \to \mathfrak{c} \triangleleft S$ under the above isomorphism, then one is a left module iff the other is.

$7 \cong 8$: By Proposition 5.18 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 8.14. □

Exercise 9.10 (Solution here). Complete the proof of Theorem 9.9 ($3 \cong 4$) 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 9.11 (Solution here). 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 9.12 (Solution here). We saw in Theorem 9.9 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 4.64 that for any function $f\colon A \to B$, 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 9.13. Let $\mathscr{C} = (\mathfrak{c}, \epsilon, \delta)$ be a comonoid and suppose that $\rho\colon m \to m \triangleleft \mathfrak{c}$ is a right $\mathscr{C}$-module. Then for any set A and function $f\colon A \to m \triangleleft 1$, the polynomial f^*m has an induced right module structure ρ_f fitting into the commutative square below:

$$\begin{tikzcd}
f^*m \arrow[r, "\rho_f"] \arrow[d] & f^*m \triangleleft \mathfrak{c} \arrow[d] \\
m \arrow[r, "\rho"'] & m \triangleleft \mathfrak{c}
\end{tikzcd}$$

Proof. The pullback diagram to the left defines $f^*(m)$ and that to the right is its composition with $\mathfrak{c}$

$$\begin{tikzcd}
f^*m \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & m \arrow[d] \\
A \arrow[r, "f"'] & m \triangleleft 1
\end{tikzcd}
\qquad
\begin{tikzcd}
f^*m \triangleleft \mathfrak{c} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & m \triangleleft \mathfrak{c} \arrow[d] \\
A \arrow[r, "f"'] & m \triangleleft 1
\end{tikzcd}$$

which is again a pullback by Theorem 5.44 and Exercise 5.17. Now the map $\rho\colon m \to m \triangleleft \mathfrak{c}$ induces a map $\rho_f\colon f^*(m) \to f^*(m) \triangleleft c$; 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 9.14. □

Exercise 9.14 (Solution here). Let $\mathfrak{c}, \epsilon, \delta)$, $\rho\colon m \to m \triangleleft \mathfrak{c}$, and $f\colon A \to m \triangleleft 1$ be as in Proposition 9.13. 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 9.15 (Polynomial functors of many variables)**.** GK definition.

Proposition 9.16. The bicategory of polynomial functors in the sense of [**GK**] is precisely that of bimodules between discrete categories.

Proof. ** □

Example 9.17 (Cellular automata). In Example 3.52 and Exercise 3.53 we briefly discussed cellular automata; here we will discuss another way that cellular automata show up, this time in terms of bimodules.

Suppose that $\mathrm{src}, \mathrm{tgt}\colon A \rightrightarrows V$ is a graph, and consider the polynomial

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

so that positions are vertices and directions are emanating arrows. It carries a natural bimodule structure

$$\begin{tikzcd} Vy & Vy \arrow[l, "g"'] \end{tikzcd}$$

where the right structure map uses tgt; see Exercise 9.18 for details. A bimodule

$$\begin{tikzcd} Vy & 0 \arrow[l, "T"'] \end{tikzcd}$$

can be identified with a functor $T\colon V \to \mathbf{Set}$, i.e. it assigns to each vertex $v \in V$ a set. Let's call $T(v)$ the color set at v; for many cellular automata we will put $T(v) \cong 2$ for each v.

Then a cellular automata on g with color sets T is given by a map

$$\begin{tikzcd} Vy & Vy \arrow[l, "g"'] & 0 \arrow[l, "T"'] \arrow[ll, bend left=40, "T"] \arrow[ll, phantom, bend left=20, "\Downarrow\alpha"] \end{tikzcd}$$

Indeed, for every vertex $v \in V$ the map α gives a function

$$\prod_{\mathrm{src}(a)=v} T(\mathrm{tgt}(a)) \xrightarrow{\alpha_v} T(v),$$

which we call the *update* function. In other words, given the current color at the target of each arrow emanating from v, the function α_v returns a new color at v.

Note that if $V \in {}_{Vy}\mathbf{Mod}_0$ is the terminal object, then the composite $Vy \xleftarrow{g} Vy \xleftarrow{V} 0$ is again V.

Exercise 9.18 (Solution here). Let src, tgt: $A \rightrightarrows V$ and g and T be as in Example 9.17.

1. Give the structure map $\lambda\colon g \to Vy \triangleleft g$
2. Give the structure map $\rho\colon g \to g \triangleleft Vy$.
3. Give the set T and the structure map $T \to Vy \triangleleft T$ corresponding to the functor $V \to$ **Set** that assigns $\mathsf{2}$ to each vertex. ◊

Example 9.19 (Running a cellular automaton). Let g be a graph on vertex set V, let T assign a color set to each $v \in V$, and let α be the update function for a cellular automaton. As in Example 9.17, this is all given by a diagram

$$\begin{tikzcd}
Vy & Vy \arrow[l, "g"'] & 0 \arrow[l, "T"'] \arrow[ll, bend left=50, "T"] \\
& \Downarrow\alpha &
\end{tikzcd}$$

To run the cellular automaton, one simply chooses a starting color in each vertex. We call this an initialization; it is given by a map of bimodules

$$\begin{tikzcd}
Vy & Vy \arrow[l, bend right=50, "Vy"'] \arrow[l, bend left=50, "T"] \arrow[l, phantom, "\Downarrow\sigma"]
\end{tikzcd} \tag{9.20}$$

Now to run the cellular automaton on that initialization for $k \in \mathbb{N}$ steps is given by the composite

$$\begin{tikzcd}
Vy \arrow[d, equal] & \cdots \arrow[l, "g"'] & Vy \arrow[l, "g"'] \arrow[d, equal] & Vy \arrow[l, "g"'] \arrow[d, equal] & 0 \arrow[l, "g"'] \arrow[d, equal] \arrow[llll, bend right=30, "V"'] \arrow[dl, phantom, "\Downarrow\sigma"] \\
Vy \arrow[d, equal] & \cdots \arrow[l, "g"'] & Vy \arrow[l, "g"'] \arrow[d, equal] & Vy \arrow[l, "g"'] \arrow[d, phantom, "\Downarrow\alpha"] & 0 \arrow[l, "T"'] \arrow[d, equal] \\
Vy \arrow[d, equal] & \cdots \arrow[l, "g"'] & Vy \arrow[l, "g"'] \arrow[d, Rightarrow, "\alpha\circ\cdots\circ\alpha"] & & 0 \arrow[ll, "T"'] \arrow[d, equal] \\
Vy & & & & 0 \arrow[llll, "T"]
\end{tikzcd}$$

Exercise 9.21 (Solution here). Explain why (9.20) models an initialization, i.e. a way to choose a starting color in each vertex. ◊

Proposition 9.22. 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 $(y^G \triangleleft \delta)\colon (y^G \triangleleft \mathfrak{c}) \to (y^G \triangleleft \mathfrak{c} \triangleleft \mathfrak{c})$. It satisfies the unitality and associativity laws because $\mathfrak{c}$ does. □

We can think of elements of G as "generators". Then if $i'\colon G \to \mathfrak{c} \triangleleft 1$ 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 9.23. Functions $i'\colon G \to \mathfrak{c} \triangleleft 1$ 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 9.13. 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 9.9. The elements of this $\mathcal{C}$-set are pairs (g, f), where $g \in G$ is a generator and $f\colon i'(g) \to \operatorname{cod}(f)$ is a morphism in $\mathcal{C}$ emanating from $i'(g)$. It is easy to see that the module structure induced by Proposition 9.22 is indeed the free one. □

Exercise 9.24 (Solution here). 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 5.51 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

$$\begin{tikzcd}
y^{\mathfrak{c}[i]} \arrow[r, "\delta^i"] \arrow[d, "\varphi"'] \arrow[dr, phantom, very near start, "\lrcorner"] & y^{\mathfrak{c}[i]} \triangleleft \mathfrak{c} \arrow[d, "\text{cart}"] \\
\mathfrak{c} \arrow[r, "\delta"'] & \mathfrak{c} \triangleleft \mathfrak{c}
\end{tikzcd}$$

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 $\delta\colon \mathfrak{c} \to \mathfrak{c} \triangleleft \mathfrak{c}$ 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 $f\colon i \to i'$ 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 $y^{\mathfrak{c}[f]}\colon y^{\mathfrak{c}[i]} \to y^{\mathfrak{c}[i']}$.

Exercise 9.25 (Solution here).

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 9.26 (Yoneda)**.** Let $\mathcal{C}$ be a category. We refer to the above functor $y^{\mathfrak{c}[-]} \colon \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ as the *Yoneda* functor.

Proposition 9.27. The Yoneda functor $y^{\mathfrak{c}[-]} \colon \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ is fully faithful.

Proof. ** □

Proposition 9.28. 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 $y^{\mathrm{Ob}(\mathcal{C})}$

$$\begin{tikzcd} y \arrow[r, "F"] \arrow[rr, bend right, "\lim F"'] & \mathscr{C} \arrow[r, "y^{\mathrm{Ob}(\mathcal{C})}"] & y \end{tikzcd}$$

Proposition 9.29. 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 9.30. 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 9.29 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. □

9.1 Morphisms between bimodules

Proposition 9.31. For any two categories $\mathfrak{c}, \mathfrak{d}$, the category ${}_{\mathfrak{d}}\mathbf{Mod}_{\mathfrak{c}}$ of bimodules between them 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 $\mathfrak{d} \xleftarrow{m}\!\triangleleft\; \mathfrak{c}$ and $\mathfrak{d} \xleftarrow{n}\!\triangleleft\; \mathfrak{c}$, the coproduct has carrier $m + n$, and the product has carrier $m \times_{d \triangleleft 1} n$. **Finish** □

Proposition 9.32. For any two categories $\mathfrak{c}, \mathfrak{d}$, the category ${}_{\mathfrak{d}}\mathbf{Mod}_{\mathfrak{c}}$ has all limits and is extensive.

Proof. ** □

9.1.1 Adjoint bimodules

Exercise 9.33 (Solution here). Recall that there is a unique $(0, \mathcal{C})$-bimodule, namely $0 \xleftarrow{0}\!\triangleleft\; \mathcal{C}$.

1. Show that 0 has a left adjoint $\mathcal{C} \xleftarrow{\quad}\!\triangleleft\; 0$; what is its carrier?
2. Show that 0 has a right adjoint $\mathcal{C} \xleftarrow{\quad}\!\triangleleft\; 0$; what is its carrier? ◊

9.2 Bimodules as data migration functors

Proposition 9.34. Let $\mathcal{C}$ and $\mathcal{D}$ be categories; the following conditions on a functor $F \colon \mathcal{C}\text{-}\mathbf{Set} \to \mathcal{D}\text{-}\mathbf{Set}$ 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.

Warning 9.35. **Cat** is not locally cartesian closed, so our notion of $\Sigma\Pi\Delta$ is not the one in **[gambino2013polynomial]**.

Consider a category $\mathcal{C}$ and a bimodule $\mathcal{C} \xleftarrow{p}\!\triangleleft\; \mathcal{C}$ from it to itself. Since $\mathcal{C}$-**Set** is locally cartesian closed, one could ask whether p arises from the bridge diagram

$$\begin{tikzcd}
& E \arrow[r] \arrow[dl] & B \arrow[dr] & \\
1 & & & 1
\end{tikzcd}$$

for a map of copresheaves $E \to B$. It may not!

As a counterexample, consider the walking arrow category $\boxed{\bullet \to \bullet}$. The functor sending the copresheaf $A \to B$ to $\emptyset \to A$ is a prafunctor, but it is not representable by a map of copresheaves; see Exercise 9.36

Exercise 9.36 (Solution here). Prove the counterexample from Warning 9.35. ◊

Definition 9.37. Pra-functors Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *pra-functor* (also called a *parametric right adjoint functor*) $\mathcal{C}$-**Set** → $\mathcal{D}$-**Set** is one satisfying any of the conditions of Proposition 9.34

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 9.38. The 2-functor Copsh: $\mathbf{Mod} \to \mathbf{Cat}$ 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 9.9.

Finish □

Exercise 9.39 (Solution here). 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.

◊

9.3 Monoidal operations

Theorem 9.40. 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 9.41. 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 9.42. 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 9.40 with $\mathcal{D} = y$ and the equivalence ${}_{y}\mathbf{Mod}_{y} \cong \mathbf{Poly}$. ☐

Proposition 9.43. The monoidal operation $+$ is a coproduct in $\mathbf{Mod}$. 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 9.41:

$$ \begin{aligned} {}_{\mathcal{C}+\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} &\cong \mathbf{Cat}(\mathcal{C}+\mathcal{D},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \\ &\cong \mathbf{Cat}(\mathcal{C},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \times \mathbf{Cat}(\mathcal{D},\ {}_{y}\mathbf{Mod}_{\mathcal{E}}) \\ &\cong {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{E}} \times {}_{\mathcal{D}}\mathbf{Mod}_{\mathcal{E}} \square \end{aligned} $$

In fact, $+$ is almost a biproduct in $\mathbf{Mod}$, except the required equivalence is replaced by an adjunction.

Proposition 9.44. For any categories $\mathcal{C}, \mathcal{D}, \mathcal{E}$ there is an adjunction

$$\begin{tikzcd} {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}+\mathcal{E}} \arrow[r, shift left=1.5ex] \arrow[r, phantom, "\Rightarrow"] & {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{E}} \arrow[l, shift left=1.5ex] \end{tikzcd}$$

Proof. By Corollary 9.41, it suffices to show that there is an adjunction

$$\begin{tikzcd} {}_{y}\mathbf{Mod}_{\mathcal{D}+\mathcal{E}} \arrow[r, shift left=1.5ex] \arrow[r, phantom, "\Rightarrow"] & {}_{y}\mathbf{Mod}_{\mathcal{D}} \times {}_{y}\mathbf{Mod}_{\mathcal{E}} \arrow[l, shift left=1.5ex] \end{tikzcd} .$$

** ☐

Proposition 9.45. If $\mathfrak{c} \overset{m_1}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$ and $\mathfrak{c} \overset{m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$ are bimodules, then so are

$$\mathfrak{c} \overset{m_1+m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d} \qquad \text{and} \qquad \mathfrak{c} \overset{m_1\times_{\mathfrak{c}(1)}m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$$

Proposition 9.46. If $\mathfrak{c}_1 \overset{m_1}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}_1$ and $\mathfrak{c}_2 \overset{m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}_2$ are bimodules, then so are

$$\mathfrak{c}_1 + \mathfrak{c}_2 \overset{m_1+m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}_1 + \mathfrak{d}_2 \qquad \text{and} \qquad \mathfrak{c}_1 \otimes \mathfrak{c}_2 \overset{m_1\otimes m_2}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}_1 \otimes \mathfrak{d}_2$$

9.4 Composing bimodules

We denote the composite of $\mathfrak{e} \overset{n}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d} \overset{m}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{c}$ by

$$\mathfrak{e} \overset{n\triangleleft_{\mathfrak{d}}m}{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{c}.$$

Recall the Yoneda functor $y^{\mathfrak{c}[-]} \colon \mathcal{C} \to {}_y\mathbf{Mod}_{\mathcal{C}}$ from Definition 9.26. Using bimodule composition, we obtain the composite functor

$$\mathcal{C} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \xrightarrow{y^{\mathfrak{c}[-]}\times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}}} {}_y\mathbf{Mod}_{\mathcal{C}} \times {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \overset{[}{\to} \triangleleft_{\mathcal{C}}]\, {}_y\mathbf{Mod}_{\mathcal{D}}.$$

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 9.47. 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 9.48. 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 9.49. 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 9.50 (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 $\mathcal{C} \overset{\check{\mathfrak{c}}}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} y$.

Proof. ** ☐

Proposition 9.51. The functor $\eta_{\mathcal{C}} \colon \mathcal{C} \to \mathbf{Set}[\mathcal{C}]$ corresponding to the unit bimodule $\mathfrak{c} \overset{\mathfrak{c}}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathfrak{c}$ sends each object $i \in \mathfrak{c}(1)$ to the bimodule $y \overset{y^{\mathfrak{c}[i]}}{\triangleleft\!\!\longrightarrow\!\!\triangleleft} \mathfrak{c}$.

Proposition 9.52. The functor $(\mathcal{C}\text{-}\mathbf{Set})^{\mathrm{op}} \to \mathbf{Set}[\mathcal{C}]$ has a left adjoint,

$$\begin{tikzcd}
(\mathcal{C}\text{-}\mathbf{Set})^{\mathrm{op}} \arrow[r, shift left=2, "y^{-}"] \arrow[r, phantom, "\Leftarrow"] & \mathbf{Set}[\mathcal{C}] \arrow[l, shift left=2, "\Gamma"]
\end{tikzcd} .$$

Proof. ** ☐

Theorem 9.53. For any categories $\mathcal{C}, \mathcal{D}$, there is an adjunction

$$\begin{tikzcd}
({}_{\mathcal{C}^{\mathrm{op}} \times \mathcal{D}}\mathbf{Mod}_0)^{\mathrm{op}} \arrow[r, shift left=2] \arrow[r, phantom, "\Rightarrow"] & {}_{\mathcal{C}}\mathbf{Mod}_{\mathcal{D}} \arrow[l, shift left=2]
\end{tikzcd}$$

such that the left adjoint is a fully faithful inclusion of profunctors into bimodules.

Proposition 9.54. For any category $\mathcal{C}$, the carrier functor creates limits.

Proof. ** ☐

Theorem 9.55. 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 9.56. 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 9.57. For any categories $\mathcal{C}$ and $\mathcal{D}$ and $(\mathcal{D}, \mathcal{C})$-bimodule $\mathfrak{d} \overset{m}{\triangleleft\!\!\longleftarrow\!\!\triangleleft} \mathfrak{c}$, we have

$$m \triangleleft_{\mathfrak{c}} \mathfrak{c} \cong m$$

where $\mathfrak{c} \overset{\mathfrak{c}}{\triangleleft\!\!\longleftarrow\!\!\triangleleft} y$ is the regular left $\mathfrak{c}$-module.

Theorem 9.58. For any category $\mathfrak{c}$, there is an adjunction

$$\begin{tikzcd} \mathbf{Cat}^{\sharp}/\mathfrak{c} \arrow[r, shift left, "\Rightarrow"'] & {}_{\mathfrak{c}}\mathbf{Mod}_0 \arrow[l, shift left, "\int"] \end{tikzcd}$$

where $\int$ is the usual category-of-elements ("Grothendieck") construction.

Moreover, $\int$ is fully faithful, and the unit map on $\mathbf{Cat}^{\sharp}/c$ sends $d \nrightarrow c$ to its vertical-cartesian factorization; see Proposition 6.98.

9.5 Exercise solutions

Chapter 10

The framed bicategory

A type of categorical structure called framed bicategory—also known as a proarrow equipment—gives a nice way to organize what's going on with comonoids, cofunctors, bimodules, and maps between them. A framed bicategory is the type of double category, so it has objects, vertical morphisms, horizontal morphisms, and 2-cells. Thus we will discuss a framed bicategory $\mathbb{P}$ for which the objects are categories, the vertical morphisms are cofunctors, and the horizontal morphisms are parametric right adjoints.

10.1 Adjoint bimodules

Proposition 10.1. A functor $F\colon \mathcal{C} \to \mathcal{D}$ gives rise to a right adjoint bimodule (= left adjoint pra) $\mathfrak{c} \overset{\Delta_F}{\lhd\!\!\longleftarrow\!\!\lhd} \mathfrak{d}$.

Proof. The carriers of $\mathfrak{c} \overset{\Delta_F}{\lhd\!\!\longleftarrow\!\!\lhd} \mathfrak{d}$ and its right adjoint $\mathfrak{d} \overset{\Pi_F}{\lhd\!\!\longleftarrow\!\!\lhd} \mathfrak{c}$ are the polynomials

$$\sum_{i \in \mathfrak{c}(1)} y^{Fi} \qquad \text{and} \qquad \sum_{j \in \mathfrak{d}(1)} y^{\sum_{j \to j'}(Fi = j')}$$

respectively. One may recognize the exponent of the latter, namely $\sum_{j \to j'}(Fi = j')$ as the set of objects in the comma category $(j \downarrow F)$, which shows up in the usual conical limit formula for the right Kan extension Π_F. □

Proposition 10.2. A cofunctor $\varphi\colon \mathcal{C} \nrightarrow \mathcal{D}$ gives rise to a left adjoint bimodule (= right adjoint pra) $\mathfrak{c} \overset{\widehat{\varphi}}{\lhd\!\!\longleftarrow\!\!\lhd} \mathfrak{d}$; we denote its right adjoint bimodule (= left adjoint pra) by $\check{\varphi}$,

$$\begin{tikzcd} \mathfrak{c} \arrow[r, bend left, "\check{\varphi}"] \arrow[r, phantom, "\Rightarrow"] & \mathfrak{d} \arrow[l, bend left, "\widehat{\varphi}"] \end{tikzcd} \tag{10.3}$$

Note that the adjunction notation in (10.3) is unambiguous: if you read $\mathfrak{c} \mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$ in the bimodule direction—i.e. as a map from $\mathfrak{c}$ to $\mathfrak{d}$—then the notation indicates that $\widehat{\varphi}$ is the left adjoint. If instead you read $\mathfrak{c} \mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathfrak{d}$ in the pra direction—i.e. as a map from $\mathfrak{d}$ to $\mathfrak{c}$—then the notation indicates that $\widehat{\varphi}$ is the right adjoint.

Proof. ** ☐

Proposition 10.4. 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 10.5 (Types on database tables). For a category $\mathcal{C}$, a functor $\mathcal{C} \to$ **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 $T\colon \operatorname{Ob}(\mathcal{C}) \to \mathbf{Set}$, i.e. $\operatorname{Ob}(\mathcal{C}) \overset{T}{\mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft}} 0$. We also have the canonical cofunctor $o\colon \mathcal{C} \nrightarrow \operatorname{Ob}(\mathcal{C})$. Now given an arbitrary instance $\mathcal{C} \overset{I}{\mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft}} 0$, there are two things we could do. We could push it forward along the left adjoint prafunctor $\operatorname{Ob}(\mathcal{C}) \mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \mathcal{C}$ and then map it to T.

Perhaps better would be to compose T with the right adjoint prafunctor $\mathcal{C} \mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft} \operatorname{Ob}(\mathcal{C}) \overset{T}{\mathrel{\triangleleft\!\!\relbar\!\!\relbar\!\!\triangleleft}} 0$ 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. $X = \check{o} \triangleleft_{\operatorname{Ob}\mathcal{C}} T$) 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 10.6. 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 10.7. 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, $\widehat{\varphi}$ has an extra adjoint

$$\begin{tikzcd}
\mathfrak{c} \arrow[r, bend left=60, "\check{\varphi}"] \arrow[r, bend right=60, "\widetilde{\varphi}"'] \arrow[r, phantom, shift left=3, "\Rightarrow"] \arrow[r, phantom, shift right=3, "\Leftarrow"] & \mathfrak{d} \arrow[l, "\widehat{\varphi}" description]
\end{tikzcd}$$

Proof. ** ☐

Proposition 10.8. Every bimodule $\mathfrak{c} \xleftarrow{S}\!\lhd\ 0$, is isomorphic to the composite

$$\mathfrak{c} \xleftarrow{Sy}\!\lhd\ y \xleftarrow{1}\!\lhd\ 0$$

for a bimodule $\mathfrak{c} \xleftarrow{Sy}\!\lhd\ y$ with carrier $Sy \in$ **Poly**.

Moreover, Sy is adjoint to a bimodule $y \xleftarrow{y^S}\!\lhd\ \mathfrak{c}$ with carrier $y^S \in$ **Poly**.

Proof. **Construct this using Theorem 9.58 and Corollary 10.7.** □

Now that we have the double category $\mathbb{P}$, we can consider the squares between cofunctors

$$\begin{tikzcd}
\mathfrak{c} \arrow[r, "\mathfrak{c}"] \arrow[d, "f"'] \arrow[dr, phantom, "\Downarrow"] & \mathfrak{c} \arrow[d, "g"] \\
\mathfrak{d} \arrow[r, "\mathfrak{d}"'] & \mathfrak{d}
\end{tikzcd}$$

Definition 10.9 (Natural co-transformations)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let $\varphi, \varphi' \colon \mathcal{C} \nrightarrow \mathcal{D}$ be cofunctors. A *natural cotransformation* $\alpha \colon \varphi \to \varphi'$ between them consists of

1. a function $\alpha_1 \colon \operatorname{Ob}(\mathcal{C}) \to \operatorname{Ob}(\mathcal{C})$
2. for each $i \in \operatorname{Ob}(\mathcal{C})$, a morphism $\alpha_i^\sharp \colon i \to \alpha_1(i)$

satisfying two conditions for every $i \in \operatorname{Ob}(\mathcal{C})$. Namely, letting $i' := \alpha_1(i)$ we require:

1. $\varphi_1(i) = \varphi_1'(i')$, call it j, and
2. for each $f \in \mathcal{D}[j]$ we have $\alpha_1(\operatorname{cod} \varphi_i^\sharp(f)) = \operatorname{cod}(\varphi')_{i'}^\sharp(f)$ and the diagram below commutes in $\mathcal{C}$:

$$\begin{tikzcd}
i \arrow[r, "\alpha_i^\sharp"] \arrow[d, "\varphi_i^\sharp(f)"'] & i' \arrow[d, "(\varphi')_{i'}^\sharp(f)"] \\
\operatorname{cod} \varphi_i^\sharp(f) \arrow[r, "\alpha^\sharp_{\operatorname{cod} \varphi_i^\sharp(f)}"'] & \operatorname{cod}(\varphi')_{i'}^\sharp(f)
\end{tikzcd}$$

10.2 Monoidal structures on $\mathbb{P}$

For any bimodules $\mathfrak{c}_1 \xleftarrow{m_1}\!\lhd\ \mathfrak{d}_1$ and $\mathfrak{c}_7 \xleftarrow{m_2}\!\lhd\ \mathfrak{d}_7$, we can construct a new bimodule in two ways:

$$\mathfrak{c}_1 + \mathfrak{c}_2 \xleftarrow{m_1 + m_2}\!\lhd\ \mathfrak{d}_1 + \mathfrak{d}_2 \qquad \text{and} \qquad \mathfrak{c}_1 \otimes \mathfrak{c}_7 \xleftarrow{m_1 \otimes m_2}\!\lhd\ \mathfrak{d}_1 \otimes \mathfrak{d}_2$$

One can think of this quite easily in terms of prafunctors: given prafunctors $\mathcal{D}_1$-**Set** $\to$ $\mathcal{C}_1$-**Set** and $\mathcal{D}_2$-**Set** $\to$ $\mathcal{C}_2$-**Set**, one obtains prafunctors between the coproduct categories and the product categories

$$(\mathcal{D}_1 + \mathcal{D}_2)\text{-}\mathbf{Set} \to (\mathcal{C}_1 + \mathcal{C}_2)\text{-}\mathbf{Set}$$

In this section we will explain how this looks in terms of bimodules, e.g. the respective bimodule structures on $m_1 + m_2$ and $m_1 \otimes m_2$, and show that each of these constructions satisfies all the conditions to be a monoidal structure on the framed bicategory $\mathbb{P}$.

10.3 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 $\mathcal{T}_p$-**Set** of dynamical systems with interface p, in terms of p?
3. How does the logic of the topos $\mathcal{T}_p$ 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 $\mathcal{T}_p \to \mathcal{T}_q$ 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 $p \mapsto \mathcal{T}_p$?
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 $\mathbf{Set} \to \mathbf{Set}$, what happens if we replace sets with homotopy types: how much goes through? Is anything improved?
10. Are there any functors $\mathbf{Set} \to \mathbf{Set}$ 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"?

10.4 Exercise solutions

Bibliography

[GK12] Nicola Gambino and Joachim Kock. “Polynomial functors and polynomial monads”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154.1 (Sept. 2012), pp. 153–192 (cit. on p. 149).

[Hed18] Jules Hedges. *Limits of bimorphic lenses*. 2018. eprint: `arXiv:1808.05545` (cit. on p. 48).

[Kel74] G Max Kelly. “Doctrinal adjunction”. In: *Category seminar*. Springer. 1974, pp. 257–280 (cit. on p. 207).

[Lac10] Stephen Lack. “Note on the construction of free monoids”. In: *Applied categorical structures* 18.1 (2010), pp. 17–29 (cit. on p. 203).

[Mac98] Saunders Mac Lane. *Categories for the working mathematician*. 2nd ed. Graduate Texts in Mathematics 5. New York: Springer-Verlag, 1998 (cit. on p. 47).

[MM92] Saunders MacLane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 1992 (cit. on p. 25).

[nLa18] Contributors To nLab. *Created limit — nLab*. 2018. URL: `https://ncatlab.org/nlab/show/created+limit` (cit. on p. 186).

[nLa21] nLab authors. *category of monoids*. `http://ncatlab.org/nlab/show/category+of+monoids`. Revision 28. Jan. 2021 (cit. on p. 47).

[Par69] Robert Paré. “Absolute coequalizers”. In: *Category Theory, Homology Theory and their Applications I*. Ed. by Peter J. Hilton. Berlin, Heidelberg: Springer, 1969, pp. 132–145. ISBN: 978-3-540-36095-7 (cit. on p. 186).

[Por19] Hans-E Porst. “Colimits of monoids”. In: *Theory and Applications of Categories* 34.17 (2019), pp. 456–467 (cit. on p. 186).

[Shu08] Michael Shulman. “Framed bicategories and monoidal fibrations”. In: *Theory and Applications of Categories* 20 (2008), Paper No. 18, 650–738 (cit. on pp. 13, 116, 121).