Applications of multivariate polynomials

David Spivak

March 1, 2024

Abstract

Coalgebras on one-variable polynomials p correspond to generalized Moore machines. But what happens when you pass to the double category $\mathbb{C}\mathbf{at}^\sharp$ of comonoids and bicomodules in **Poly**? It has a full subcategory whose objects are sets I, J : **Set** and for which a horizontal morphism is a system of I-many polynomials in J-many variables. I'll explain how multivariable polynomials give rise to "indexed types", mode-dependent dynamics, and mutually-recursive data types.

One thing we spent some time on, especially in the second week of the Poly@Work workshop, was bicomodules. The comonoids in $(\mathbf{Poly}, y, \triangleleft)$ are categories, the comonoid homomorphisms are cofunctors, and the bicomodules

$$c \xleftarrow{\ p\ } d \tag{1}$$

are "data migration functors". Our conversations helped me recall some ideas I'd had a few years ago, fill in some missing details, get a sense of what they might be useful for, and learn more about computer science applications and names, such as Generalized Algebraic Data Types, and indexed types.

In this note, I'll explain how multivariate polynomials really bake in mode-dependence, a kind of typing discipline, for Moore machines. I'll also explain how they allow for mutually-recursive data types (the language generated by a context-free grammar).

Thanks to Harrison Grodin, Reed Mullanix, Nelson Niu, Nate Osgood, and everyone else at the Poly workshop for helpful and fun conversations!

Systems of multivariate polynomials as bicomodules

Given polynomial comonoids (categories) c and d, a bicomodule (1) between them consists of a polynomial p together with maps

$$c \triangleleft p \xleftarrow{\lambda} p \xrightarrow{\rho} p \triangleleft d$$

satisfying five equations: unitality and associativity on each side, as well as a compatibility equation. As alluded to above, they correspond to functors d-**Set** $\to$ c-**Set**, and generalize profunctors.

But they are a lot of data, and sometimes difficult to think about. Luckily, they simplify a great deal when c, d are discrete categories, i.e. comonoids of the form $c = Cy$ and $d = Dy$. Then a bicomodule p as above can be identified with C-many polynomials in D-many variables. For example, if $C = 2$ and $D = 3$, then we can think of p as somehow decomposable as

$$\begin{aligned} p_1 &= y_1^4 y_2^3 + 17 y_1 y^2 y_3 + 2 y_3^9 \\ p_2 &= y_2^7 y_3^3 + 6 \end{aligned} \tag{2}$$

This decomposition is encoded by the bicomodule structure maps

$$2y \triangleleft p \xleftarrow{\lambda} p \xrightarrow{\rho} p \triangleleft 3y.$$

Indeed, the map λ encodes a sum-decomposition of positions $p(1) = p_1(1) + p_2(1)$, and for each position $P : p(1)$, the map ρ encodes a splitting up of exponents into $p[P] = p[P]_1 + p[P]_2 + p[P]_3$. So the bicomodule corresponding to the polynomials from (2) would be of the form

$$2y \xleftarrow{y^7+17y4+2y^9+y^{10}+6} 3y$$

One interesting fact[1] is that a comonad in $\mathbb{C}\mathbf{at}^\sharp$, i.e. a bicomodule $c \xleftarrow{d} c$ equipped with maps

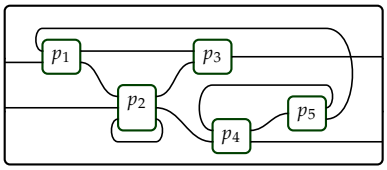
(3)

satisfying counitality and comultiplication can be identified with a cofunctor $d \twoheadrightarrow c$. That is, it looks complicated, but it's pretty easy.

So, for example, if we have a comonad of the form $3y \xleftarrow{d} 3y$, then it is the same thing as a category d equipped with a cofunctor to the three object category 3, which is the same as a way of labeling each object in d with an element in the set {'1', '2', '3'}.

Moore machines

Moore machines are ways of taking inputs and producing outputs, all while keeping an internal state. For example, a machine that takes in a stream of integers and produces a running total is a kind of Moore machine. We can wire together Moore machines to get new Moore machines.



(4)

An *uninitialized Moore machine* with input type A and output type B consists of a set S, called the set of *states*, and a coalgebra

$$S \to By^A \triangleleft S = B \times S^A$$

[1]This results from general theory about the comonoids-and-bicomodules construction: Taking **Comod** of **Comod** always has this sort of behavior.

This could also be written as a polynomial map

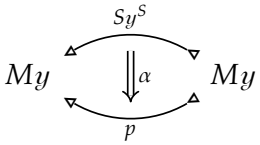
$$Sy^S \to By^A.$$

To understand what it means, we unfold it into two functions $r\colon S \to B$, which we call the *readout*, and $u\colon S \times A \to S$, which we call the *update*. Choosing an initial state $s_0\colon 1 \to S$ and a list $[a_0, \ldots, a_n]$, one obtains a list of outputs $[b_0, \ldots, b_n]$ by $s_{i+1} \coloneqq u(a_i, s_i)$ and $b_i \coloneqq r(b_i)$.

Sometimes one doesn't care about the states themselves, one only cares about the *behavior* of the machine, i.e. the flow chart that says what the machine is outputting now, as well as for every possible input, a new flow chart. These are the elements of the terminal coalgebra, which is also the set of positions of the cofree comonad $\mathfrak{c}_p$ on p.

This is great, but it's pretty undisciplined. Given an input $a : A$, the current state can go to any new state. What if we want to be more disciplined and decompose our set S of states into various bins, called *modes*. Then we could specify which outputs were possible in any given mode, as well as specify the mode resulting from receiving any given input.

So let M be a set, elements of which we'll call "modes", and suppose we have a function $m\colon S \to M$. As we saw in (3), this is the same thing as a comonad carried by a bicomodule of the form $My \xleftarrow{Sy^S} My$. Then a Moore machine $Sy^S \to p$ can be upgraded to a mode dependent one by realizing it as underlying a map of the form



Suppose we have a bicomodule $3y \xleftarrow{p} 3y$:

$$\begin{aligned} p_1 &= y_1^4 y_2^3 + 17 y_1 y^2 y_3 + 2 y_3^9 \\ p_2 &= y_2^7 y_3^4 + 6 \\ p_3 &= y_3^4 \end{aligned}$$

The idea now is that each $s : S$ is in some mode $m(s) : M = \{'1', '2', '3'\}$. It must be sent by α_1 to some position in $p_{m(s)}(1)$. Then that position has some directions on each variable. An input is given by picking one of those directions. Once we do, we'll have in hand the variable (and hence mode) that it was a direction on, and the map $\alpha_s^\sharp$ will send it back to a state s in that mode.

So for example, if we had a state s in mode $m(s) = 2$, it will be sent by α_1 to one of the 7 positions in p_2. If it is sent to the position labeled $y_2^7 y_3^3$, then an input is one of 11 things, each of which will result in a state update. The new state will be in mode $m(\alpha_s^\sharp(d) = 2$ if d is in the first seven directions, and will be in model 3 if d is in the second four directions.

In general, rather than thinking about Sy^S, one can think about any comonad d in its place and studying maps of bicomodules $d \to p$. But doing so, one realizes that what they're actually studying the cofree comonad $\mathfrak{c}_p$ on p, but this time where the symbol $\mathfrak{c}$ refers to the comonad taken in the category of (My, My) bicomodules.

It would be interesting to think about wiring diagrams as in (4) in this setting.

Indexed types, context free grammars, and mutually recursive data types

A context free grammar has some set S of symbols, and for each $s : S$, some set p_s of production rules, each of which involving various symbols. This can all be encoded into a multivariable polynomial of the form

$$Sy \xleftarrow{p} Sy.$$

Thanks to Reed and Harrison, I know that these are strongly related to indexed types and GADTs (generalized algebraic datatypes). The idea is to form the free monad $\mathfrak{m}_p$ on this data. For any set of literals $L\colon S \to \mathbf{Set}$ for each symbol, one can take $\mathfrak{m}_p(S)$, which in the case of context free grammars will be the resulting language. The initial algebra on p is $\mathfrak{m}_p(0)$, which we can think of as a mutually recursive data type.

Other cofree comonads and free monads in bicomodules?

We also thought about bicomodules of the form $\mathfrak{c}_{p_1} \xleftarrow{} q\mathfrak{c}_{p_2}$. For example, when $p_1 = p_2 = y$, these are bicomodules of the form

$$y^{\mathbb{N}} \xleftarrow{q} y^{\mathbb{N}}$$

The positions of q form a closed dynamical system, and for each $Q : q(1)$, the directions $q[Q]$ also form a dynamical system. For any state and next state $Q_1 \mapsto Q_2$, there is a map $q[Q_1] \leftarrow q[Q_2]$, which we see pointing in the backwards direction. One can think of this as a kind of refinement: every state in Q_1 is refined into several (or no) states in Q_2.

How should one think about—i.e. what are applications of—cofree comonads or free monads for other bicomodules of the form $c \xleftarrow{p} c$? You tell me!