Notes on a Definition of Higher Structure

Eric Finster

November 24, 2018

1 Polynomials

In this section, we review the theory of polynomials and their associated functors.

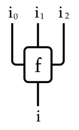
Definition 1.1. A *polynomial* P consists of

(i) A type I of *sorts*

(ii) A dependent family $\mathsf{Op} : I \to \mathsf{Type}$ assigning to each sort $i : I$ the type of *operations* with output sort i.

(iii) A dependent family $\mathsf{Param} : (i : I) \to \mathsf{Op}\, i \to (j : I) \to \mathsf{Type}$ which assigns to an operation f of output sort i and a sort j the type of *parameters* of f of input sort j

We will typically represent operations in such a polynomial using pictures like the following:



which display its output type (i) as well as it's collection of typed parameters.

Viewing a polynomial as a kind of algebraic signature, we can define the collection of *terms* generated by this signature. For historical reasons, this is called the W-*type* associated to P. Its formal definition is as follows.

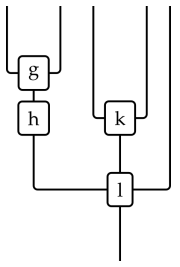
Definition 1.2. Let P be a polynomial with sorts in I. Then the W-*type* associated to P is the inductive family $\mathsf{W} : I \to \mathsf{Type}$ with constructors given by

- $\mathsf{lf} : (i : I) \to \mathsf{W}\, i$
- $\mathsf{nd} : \{i : I\} \to [\![P]\!]\, \mathsf{W}\, i \to \mathsf{W}\, i$

In what follows, I will write $\mathsf{W}\, i$ to indicate this type when it is clear from the context what polynomial is under consideration, but will add the polynomial explicitly, as in $\mathsf{W}\, P\, i$ when the situation requires it.

Remark 1.3. This definition differs slightly from the usual one in that we include the first lf constructor. With this definition, the W-type is exactly the collection of operations for the free monad on P. From the point of view of universal algebra, it corresponds to adding a single *variable* of each sort to our language of terms.

Trees generated by a polynomial will also be given a graphical representation as in the following:



Note that we typically supress the sorts which decorate the edges of such a tree.

Two definitions which will play an important role in the development to follow are the types of *leaves* and *nodes* of a given tree. Then can be defined by recursion as follows:

$$\begin{aligned}
&\mathsf{Leaf} : \{i : I\}(w : \mathsf{W}\, i) \to I \to \mathsf{Type} \\
&\mathsf{Leaf}\,(\mathsf{lf}\, i)\, j :\equiv i =_I j \\
&\mathsf{Leaf}\,(\mathsf{nd}\, i\, (f, \phi))\, j :\equiv \sum_{(k:I)} \sum_{(p:\mathsf{Param}\, f\, k)} \mathsf{Leaf}\,(\phi\, k\, p)\, j
\end{aligned}$$

Nodes may be defined in a similar manner.

$$\begin{aligned}
&\mathsf{Node} : \{i : I\}(w : \mathsf{W}\, i)(j : I) \to \mathsf{Op}\, j \to \mathsf{Type} \\
&\mathsf{Node}\,(\mathsf{lf}\, i)\, j\, g :\equiv \bot \\
&\mathsf{Node}\,(\mathsf{nd}\, \{i\}\, (f, \phi))\, j\, g :\equiv \\
&\qquad \Big((i, f) =_{\sum_{(k:I)} \mathsf{Op} k}\, (j, g)\Big) \sqcup \sum_{(k:I)} \sum_{(p:\mathsf{Param}\, f\, k)} \mathsf{Node}\, k\, (\phi\, k\, p)\, j
\end{aligned}$$

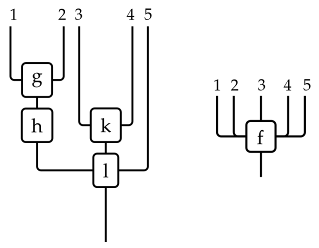
1.1 Frames and Relations

In what follows, we will be discussing *cartesian* monads. A definition which will be useful in such considerations is the following:

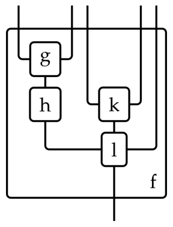
Definition 1.4. Let P be a polynomial with sorts in I. Given $i : I$, an operation $f : \mathsf{Op}\, i$ and a tree $w : \mathsf{W}\, i$, a *frame* from w to f is a family of equivalences

$$\mathsf{Frame}\, w\, f :\equiv \prod_{j:I} \mathsf{Leaf}\, w\, j \simeq \mathsf{Param}\, f\, j$$

Hence a frame is a specified isomorphism between the leaves of some P-tree and an operation of P preserving the sorts of the inputs. A first attempt at visualizing a frame might be something as follows, where the isomorphism is specified by, say, labelling the leaves and parameters in such a way as to indicate the bijection (note that the numbers have nothing to do with the sorts of each of the leaves which are left implicit here, but rather are being used to describe a chosen bijection):



However, for our purposes, it will be useful to adopt a more compact visual notation. This we can do by enlarging the box representing the operation f and "sliding it behind" the associated tree, say w, as in the following diagram:



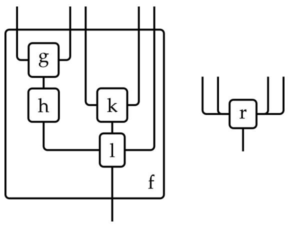
The frame has now become implicit in the diagram, corresponding to the fact that the inputs of f and the leaves of the tree w are made to coincide. One should keep in mind, however, that inside of type theory, the frame is a piece of data subject to being manipulated.

The intuition of this representation is that our diagram above is a picture of the "boundary" of a higher dimensional relation between the tree w (serving as the input) and the operation f (playing the role of the output). To make this precise, we introduce the following definition:

Definition 1.5. A *polynomial relation* for P is a type family

$$R : \{i : I\}(f : \mathsf{Op}\, i)(w : \mathsf{W}\, i)(\alpha : \mathsf{Frame}\, w\, f) \to \mathsf{Type}$$

Given a polynomial relation R and an element $r : R\, w\, f\, \alpha$, we can picture this element as in the following diagram:



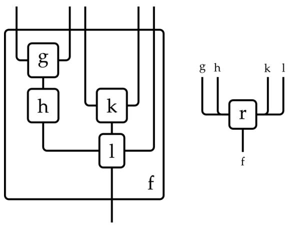
Notice how we represent the element r as if it were an operation having four inputs, just as as we did for the operations of P (such as f). In fact, this entire diagram can be see as an operation in a certain polynomial, which we introduce now.

Definition 1.6. Let P be a polynomial and R a polynomial relation on P. The *slice* of P by R, denoted $P//R$ is the polynomial defined by:

$$\mathsf{Op}(P//R)\,(i,f) := \sum_{(w:\mathsf{W}\, i)} \sum_{(\alpha:\mathsf{Frame}\, w\, f)} R\, f\, w\, \alpha$$

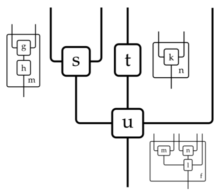
$$\mathsf{Param}(P//R)\,(i,f)(w,r)(j,g) := \mathsf{Node}\, w\, j\, g$$

Revisiting our diagram above, notice how the output type of this operation in $P//R$ is indeed labelled by an operation (f, as the case may be) of P.

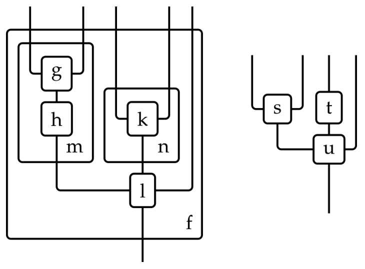


Moreover, the *inputs* of this operation are the nodes of the tree w. Here, we do not mean to indicate any bijection. Rather the nodes of w are *by definition* the parameters of this operation. (This is the advantage of our more compact notation for frames: the operation f is denoted together will all data which is proper to the polynomial P, leaving us space to denote the operations of the polynomial $P//R$.)

As any polynomial determines an associated type of trees, we may wonder: what does an element of $\mathsf{W}\,(P//R)$ look like? A first attempt might be something like the following:



where we have indicated the "lower dimensional" information by drawing it next to the node to which it is associated. And indeed, this picture in some sense accurately reflects the data provided by a $\mathsf{W}\,(P//R)$ tree. However, the requirement that each node carry a frame relating its source tree to its output operation means that all this data is in fact compatible, and we can realize the whole diagram more systematically as

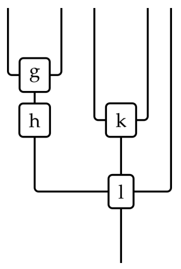


This picture then suggests some facts about our tree, which we will denote $pd : \mathsf{W}\,(P//R)\,(i, f)$. Namely, that there is a "total tree" (the one depicted in the diagram with nodes g, h, k, l) obtained by gluing together the source trees decorating each of the nodes, that, moreover, this tree admits a frame to the operation f and finally, that the leaves of pd are in fact in bijection with the nodes of the total tree.

In fact, there is such a construction, defined by a function

$$\mathsf{flatten} : \{i : I\}\{f : \mathsf{Op}\, i\} \to \mathsf{W}\,(P//R)\,(i, f) \to \mathsf{W}\, P\, i$$

Applied to the above diagram extracts the following tree:



Similarly, we can define a function

$$\mathsf{flatten\text{-}frm} : \{i : I\}\{f : \mathsf{Op}\, i\}(pd : \mathsf{W}\,(P//R)\,(i, f)) \to \mathsf{Frame}\,(\mathsf{flatten}\, pd)\, f$$

as well as an equivalence

$$\begin{aligned} \mathsf{bd\text{-}frm} :& \{i : I\}\{f : \mathsf{Op}\, i\}(pd : \mathsf{W}\,(P//R)\,(i, f))(j : I)(g : \mathsf{Op}\, j) \\ & \to \mathsf{Leaf}\,(P//R)\, pd\,(j, g) \simeq \mathsf{Node}\, P\,(\mathsf{flatten}\, pd)(j, g) \end{aligned}$$

The name bd-frm is for "Baez-Dolan frame", and, as we will see in what follows, this function will serve as a useful source of frames in the polynomial $P//R$.

I will not detail the implementation of these three functions here, but rather confine myself to a couple of remarks and use them freely in the sequel. The implementation is not difficult, essentially proceeding by induction on the tree in the most evident fashion, and using the specified frame data as necessary. It *is* however probably worth noting that the implmentation of these functions *does not depend on any additional structure on the polynomials.* That is to say, we do not need to use the "laws" for any kind of algebraic structure on our polynomials in order to implement them. Consequently, we do not suffer from any kind of coherence problem wherein the definition requires laws, which must be proven to complete the definition, which then require laws ... and so on.

1.2 Polynomial Monads

Our goal now is to show how to equip a polynomial P with a coherent monad structure. Typical axiomatizations of a monad structure on P found in the literature begin with a *biased* multiplication, i.e. binary multiplication operator on two-level trees formed from the operations of P satisfying unit and associativity axioms. As is typical for higher structures, this turns out to be slightly inconvenient as the laws then are subject to their own laws and so on.

We will therefore adopt an unbiased approach, beginning with the following definition:

Definition 1.7. Let P be a polynomial with sorts in I. A *polynomial magma* M over P is

(i) A function $\mu : \{i : I\} \to \mathsf{W}\, P\, i \to \mathsf{Op}\, P\, i$

(ii) A function $\mu_{frm} : \{i : I\}(w : \mathsf{W}\, P\, i) \to \mathsf{Frame}\, w\, (\mu\, w)$

The second function reflects the requirement that we are describing *cartesian* monads. In terms of universal algebra, we will use this multiplication operator to encode the relations present in out structure. Encoding them in this manner means restricting to relations which

(i) Are of the form "term = operation"

(ii) Preserve the number of variables on each side

So for example, writing our operations as function symbols, we are allowed relations of the form

$$f(g(x, y), z) = k(x, y, z)$$

But not

$$f(g(x, y), z) = h(x, k(y))$$

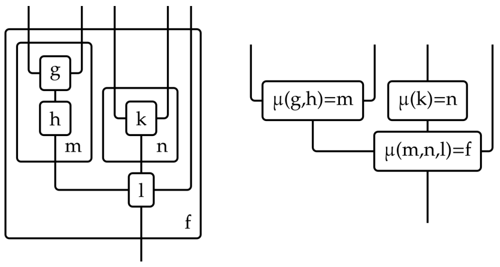
as this second equation has two operation symbols on the right and one fewer variable. Such relations can, and often will be, *implied* by our structure, they are just not primitive.

A magma M on P determines a relation simply by using the identity type. That is, we have

$$\begin{aligned} &\mathsf{MgmRel} : \mathsf{PolyMagma}\, P \to \mathsf{PolyRel}\, P \\ &\mathsf{MgmRel}\, M\, f\, w\, \alpha :\equiv (\mu\, w, \mu_{frm}\, w) = (f, \alpha) \end{aligned}$$

We will often abuse notation and write $P//M$ for the polynomial obtained by slicing P by the magma relation determined by M.

Let us revisit our depiction of an element of $\mathsf{W}\,(P//M)$ now that we have an explicit relation to consider. We might now write it as follows:



(where I have elided the extra information about the frame for space reasons). In this case, we see that the tree can be seen a sequence of multiplications performed on different subtrees of the "flattened" tree. Overall, we might write this as the equation

$$\mu(\mu(g, h), \mu(k), l) = f$$

although we are of course really interested in the structure of the whole tree as an element of $\mathsf{W}\,(P//M)\,(i, f)$.

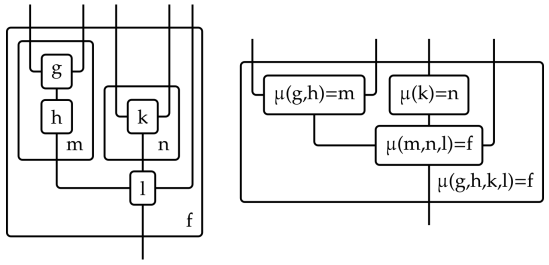
A natural question to ask is now the following: what would a magma structure for $P//M$ look like? We can already make some progress towards such a definition. That is, we can try:

$$\begin{aligned} &\mathsf{SlcMgm} : \mathsf{PolyMagma}\, P \to \mathsf{PolyMagma}(P//M) \\ \mu(\mathsf{SlcMgm}\, M)\, pd &:\equiv ((\mathsf{flatten}\, pd, \mathsf{flattn\text{-}frm}\, pd), ?) \\ \mu_{frm}(\mathsf{SlcMgm}\, M) pd &:\equiv \mathsf{bd\text{-}frm}\, pd \end{aligned}$$

But we find that we are missing a term to replace the question mark. The type of this goal is $\mathsf{MgmRel}\, M\, f\, (\mathsf{flatten}\, pd)$ which expands to

$$(\mu(\mathsf{flatten}\, pd), \mu_{frm}(\mathsf{flatten}\, pd)) = (f, \mathsf{flatten\text{-}frm}\, pd)$$

In other words, we are being asked to show that, after permorming a sequence of multiplications on subtrees of *pd* (as represented by the our tree in the slice), the result is equal to the result of applying μ to the flattened tree that *pd* determines (plus a compatibility with the associated frame). Given such an element, we could complete our picture above as follows:



from which, in terms of equations, we would deduce

$$\mu(\mu(g,h), \mu(k), l) = f = \mu(g,h,k,l)$$

Of course, we don't merely want this to happen for our fixed pasting diagram *pd*. We want this to work for *any* sequence of multiplications in M. The motivates the following defintiion (which we state at the generality of polynomial relations)

Definition 1.8. Let P be a polynomial and R a relation on P. We say that R is *subdivision invariant* if we are given a function.

$$\begin{aligned} \Psi :&\{i : I\}\{f : \mathsf{Op}\, P\, i\}(pd : \mathsf{W}\, (P//R)\, (i,f)) \\ &\to R\, f\, (\mathsf{flatten}\, pd)\, (\mathsf{flatten\text{-}frm}\, pd) \end{aligned}$$

We write $\mathsf{SubInvar}$ for the associated predicate on polynomial relations. That is

$$\begin{aligned} &\mathsf{SubInvar} : \mathsf{PolyRel}\, P \to \mathsf{Type} \\ \mathsf{SubInvar}\, R &:\equiv \{i : I\}\{f : \mathsf{Op}\, P\, i\}(pd : \mathsf{W}\, (P//R)\, (i,f)) \\ &\to R\, f\, (\mathsf{flatten}\, pd)\, (\mathsf{flatten\text{-}frm}\, pd) \end{aligned}$$

With the definition in hand, we can complete our definition of a magma structure on the slice. It now reads:

$$\begin{aligned} &\mathsf{SlcMgm} : (M : \mathsf{PolyMagma}\, P)(\Psi : \mathsf{SubInvar}\, M) \to \mathsf{PolyMagma}(P//M) \\ \mu(\mathsf{SlcMgm}\, M)\, pd &:\equiv ((\mathsf{flatten}\, pd, \mathsf{flattn\text{-}frm}\, pd), \Psi\, pd) \\ \mu_{frm}(\mathsf{SlcMgm}\, M) pd &:\equiv \mathsf{bd\text{-}frm}\, pd \end{aligned}$$

Intuitively, then, if a magma structure on P is subdivision invariant, this means we can also *multiply together instances of the associated relation.* It is not hard to check from this that if a magma M is subdivision invariant, then it is necessarily unital and associative, which is to say, equipping M with a proof of substitution invariance amounts to proving that it satisfies unit and associativity laws.

With this notion in place, we are now ready to define what a coherent monad structure on P is. First of all we have the following:

Definition 1.9. Let P be a polynomial and M a magma on P. A *coherence structure* for M is

(i) A proof $\Psi : \mathsf{SubInvar}\, M$
(ii) Coninductively, a coherence structure on $\mathsf{SlcMgm}\, M\, \Psi$

Consequently, a polynomial monad structure on P is

(i) A magma M over P
(ii) A coherence structure on M

So, the definition says that a coherence structure on M is a way to "compose relations in M" and a way to "compose relations in the slice magma determined by this composition" and so on, infinitely often.