Can you Differentiate a Polynomial?

J.R.B. Cockett

Department of Computer Science
University of Calgary
Alberta, Canada

robin@cpsc.ucalgary.ca

Halifax, June 2012

PART I: Differential Categories

PART II: Structural polynomials

One of the motivating example behind the development of Cartesian Differential Categories!

... and how examples can be very confusing.

$\times$-DIFFERENTIAL CATEGORIES

Recall to formulate $\times$-differential categories need:

(a) Left additive categories

(b) Cartesian structure in the presence of left additive structure

(c) Cartesian differential structure

Example to have in mind: vector spaces with smooth functions ....

Left-additive categories

A category $\mathbb{X}$ is a **left-additive category** in case:

- Each hom-set is a commutative monoid $(0,+)$
- $f(g+h) = (fg) + (fh)$ and $f0 = 0$
  each f is **left additive** ..

$$A \xrightarrow{f} B \underset{h}{\overset{g}{\rightrightarrows}} C$$

A map h is said to be **additive** if it also preserves the additive structure on the right $(f+g)h = (fh) + (gh)$ and $0h = 0$.

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{h} C$$

Additive maps are the exception ...

Products in left additive categories

A **Cartesian left-additive category** is a left-additive category with products such that:

- the maps π_0, π_1, and Δ are additive;
- f and g additive implies $f \times g$ additive.

Lemma

The following are equivalent:

(i) *A Cartesian left-additive category;*

(ii) *A Cartesian category $\mathbb{X}$ in which each object is equipped with a chosen commutative monoid structure*

$$(+_A : A \times A \longrightarrow A, 0_A : 1 \longrightarrow A)$$

such that $+_{A \times B} = \langle (\pi_0 \times \pi_0)+_A, (\pi_1 \times \pi_1)+_B \rangle$ *and* $0_{A \times B} = \langle 0_A, 0_B \rangle$.

The axioms for a $\times$-differential

[CD.1] $D_\times[f+g] = D_\times[f] + D_\times[g]$ and $D_\times[0] = 0$;
(operator preserves additive structure)

[CD.2] $\langle (h+k), v\rangle D_\times[f] = \langle h, v\rangle D_\times[f] + \langle k, v\rangle D_\times[f]$
(always additive in first argument);

[CD.3] $D_\times[1] = \pi_0$, $D_\times[\pi_0] = \pi_0\pi_0$, and $D_\times[\pi_1] = \pi_0\pi_1$
(coherence maps are linear);

[CD.4] $D_\times[\langle f, g\rangle] = \langle D_\times[f], D_\times[g]\rangle$ (and $D_\times[\langle\rangle] = \langle\rangle$)
(operator preserves pairing);

[CD.5] $D_\times[fg] = \langle D_\times[f], \pi_1 f\rangle D_\times[g]$ (chain rule);

[CD.6] $\langle\langle f, 0\rangle, \langle h, g\rangle\rangle D_\times[D_\times[f]] = \langle f, h\rangle D_\times[f]$
(differentials are linear in first argument);

[CD.7] $\langle\langle 0, f\rangle, \langle g, h\rangle\rangle D_\times[D_\times[f]] = \langle\langle 0, g\rangle, \langle f, h\rangle\rangle D_\times[D_\times[f]]$
(partial differentials commute);

An example Polynomials are an example:

The category Poly($\mathbb{N}$):

Objects: The natural numbers: $0, 1, 2, 3, \ldots$

Maps: $(p_1, \ldots, p_n) : m \longrightarrow n$ where $p_i \in \mathbb{N}[x_1, .., x_m]$

Composition: By substitution.

This is the Lawvere theory of commutative rigs ...

The differential is:

$$\frac{m \longrightarrow n; (x_1, .., x_m) \mapsto (p_1, .., p_n)}{(\sum_i y_i \cdot \partial_i p_1, \ldots, \sum_i y_i \cdot \partial_i p_n) : m + m \longrightarrow n}$$

Not the polynomials of this talk!

Well ... not quite!

POLYNOMIALS

A (structural) **polynomial** in any category with pullbacks is a diagram

$$\begin{tikzcd} & P \arrow[r, "u"] \arrow[dl, "v"'] & S \arrow[dr, "w"] & \\ X & & & Y \end{tikzcd}$$

in which u is **exponentiable**, that is the functor Δ_u (pulling back along u) has a right adjoint Π_u, so that $\Delta_u \vdash \Pi_u$.

Will eventually require a lextensive category ...

Ideas due to: Gambino, J. Kock, Weber, Hyland, Joyal, ...
Here I follow Gambino and Kock's development closely.

Structural polynomials

In structural polynomial

$$
\begin{tikzcd}
 & P \arrow[r, "u"] \arrow[dl, "v"'] & S \arrow[dr, "w"] & \\
X & & & Y
\end{tikzcd}
$$

think of

X as "input sort names";

P as "variable places";

S as "shapes";

Y as "output sort names".

Can encode all initial data types ....

Structural polynomial for binary trees

$$\begin{tikzcd}
 & P \arrow[r, "u"] \arrow[dl, "v"'] & S \arrow[dr, "w"] & \\
X & & & Y
\end{tikzcd}$$

Represent binary trees in Set as a polynomial:

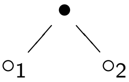
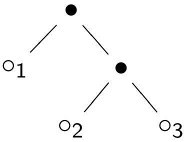
(a) There is only one input sort $X = \{A\}$;

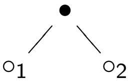
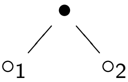
(b) S is the set of shapes of binary trees;

(c) P is the set of places where variables can occur (on the leaves) of the binary tree shapes;

(d) There is only one output sort $Y = \{\mathrm{Tree}(A)\}$

Structural polynomial for binary trees

$S = \Big\{ \circ_1,$  $,$  $\dots \Big\}$

$P = \Big\{ (1, \circ_1), (1,$  $), (2,$  $), \dots \Big\}$

The map u takes a place in a tree (a pair) to the shape of the tree.

Polynomials functors

Associated to each structural polynomial

$$\begin{tikzcd} & P \arrow[r, "u"] \arrow[dl, "v"'] & S \arrow[dr, "w"] & \\ X & & & Y \end{tikzcd}$$

is a **polynomial functor**:

$$P_{v,u,w} = \mathbb{C}/X \xrightarrow{\Delta_v} \mathbb{C}/P \xrightarrow{\Pi_u} \mathbb{C}/S \xrightarrow{\Sigma_w} \mathbb{C}/Y$$

- Δ_v is the "reindexing" or "substitution" functor (pulling back along v)
- Π_u is the "dependent product" functor (the right adjoint to $\Delta_u = u^*$)
- Σ_w is the "dependent sum" functor (given by composition $\Sigma_w(f) = fw$)

Indexed sets

$$\begin{tikzcd}
 & (Q_{v(p)})_{p:P} \arrow[dl, dotted, "(\sigma_{v(p)}(\text{-}))_{p:P}"'] \arrow[d, dotted, "\Delta_v(Q)"] & (\Pi_{p:u(p)=s} Q_{v(p)})_{s:S} \arrow[d, dotted, "\Pi_u(\Delta_v(Q))"] \arrow[dr, dotted, "\Sigma_s(\sigma_{w(s)}(\text{-}))"] & \\
(Q_x)_{x:X} \arrow[d, dotted, "Q"'] & P \arrow[dl, "v"'] \arrow[r, "u"] & S \arrow[dr, "w"] & \left(\Sigma_{s:w(s)=y}(\Pi_{p:u(p)=s} Q_{v(p)})\right)_{y:Y} \arrow[d, dotted, "\Sigma_w(\Pi_u(\Delta_v(Q)))"] \\
X & & & Y
\end{tikzcd}$$

.... structural polynomials between finite sets are (equivalent to) polynomial tuples over the rig of natural numbers.

Spans

When u is the identity we get a span:

$$\begin{tikzcd} & S \arrow[r, equal, "u"] \arrow[dl, "v"'] & S \arrow[dr, "w"] & \\ X & & & Y \end{tikzcd}$$

In a span each shape has exactly one place ...
Spans are to be thought of as a linear map ...

Spans

$$
\begin{tikzcd}
& (Q_{v(p)})_{p\in P} \arrow[r, equal] \arrow[dl, dotted] \arrow[d, dotted, "\Delta_v(Q)"] & (Q_{v(p)})_{p\in P} \arrow[d, dotted, "\Delta_v(Q)"] \arrow[dr, dotted] & \\
(Q_x)_{x\in X} \arrow[d, dotted, "Q"'] & P \arrow[r, equal] \arrow[dl, "v"'] & P \arrow[dr, "w"] & \left(\Sigma_{p:w(p)=y}(Q_{v(p)})\right)_{y\in Y} \arrow[d, dotted, "\Sigma_w(\Delta_v(Q))"] \\
X & & & Y
\end{tikzcd}
$$

Composition of polynomials

$$
\begin{tikzcd}
 & P \arrow[r, "u"] \arrow[dl, "v"] & S \arrow[dr, "w"] & & P' \arrow[r, "u'"] \arrow[dl, "v'"] & S' \arrow[dr, "w'"] & \\
X & & & Y & & & Z
\end{tikzcd}
$$

Composition of polynomials

$$
\begin{tikzcd}
 & & & P_2 \arrow[r] \arrow[dl] & S_2 \arrow[r] \arrow[dl, "\varepsilon"'] \arrow[dd] & S_2' \arrow[dd] & \\
 & & P_1 \arrow[r] \arrow[dl] & S_1 \arrow[dl] \arrow[dr] & & & \\
 & P \arrow[r, "u"] \arrow[dl, "v"] & S \arrow[dr, "w"'] & & P' \arrow[r, "u'"] \arrow[dl, "v'"] & S' \arrow[dr, "w'"'] & \\
X & & & Y & & & Z
\end{tikzcd}
$$

All squares pullbacks ... the triangle involves the counit $\varepsilon : \Delta_u(\Pi_u(A)) \longrightarrow A$.

Key Lemma for composition of polynomials

$$\begin{tikzcd}
 & {} \arrow[dr, dotted, "\Delta_\varepsilon(g)"] & & & {} \arrow[dl, dotted, "\Pi_{u^*}(\Delta_\varepsilon(g))"'] \\
 & & \Delta_u(\Pi_u(f))_0 \arrow[r, "u^*"] \arrow[dl, "\varepsilon"'] \arrow[dd, "\Delta_u(\Pi_u(f))"] & \Pi_u(f)_0 \arrow[dd, "\Pi_u(f)=h"] & \\
{} \arrow[r, dotted, "g"] & A \arrow[dr, "f"'] & & & \\
 & & P \arrow[r, "u"'] & S & \\
 & {} \arrow[ur, dotted, "\Sigma_f(g)=fg"'] & & & {} \arrow[ul, dotted, "\Pi_{u^*}(\Delta_\varepsilon(g))h"', "\Pi_u(fg)"]
\end{tikzcd}$$

Lemma

$$\Pi_u(\Sigma_f(g)) = \Sigma_h(\Pi_{u^*}(\Delta_\varepsilon(g)))$$

Key Lemma for composition of polynomials

Expresses the distributive law!

$$
\begin{tikzcd}
 & \{(x,z)_0,(y,z)_0,(x,z)_1,(y,z)_1\} \arrow[r] \arrow[dl] & \{(x,z),(y,z)\} \arrow[dd] \\
(\{x,y\}_0,\{z\}_1) \arrow[dr] & & \\
 & \{0,1\} \arrow[r] & \{\star\}
\end{tikzcd}
$$

$$(x+y)\times z = x\times z + y\times z$$

Composition of polynomials

For proof also need Beck-Chevalley ... given the pullback squares

$$\begin{tikzcd} A \arrow[r, "f'"] \arrow[d, "g'"'] & B \arrow[d, "g"] \\ A' \arrow[r, "f"'] & B' \end{tikzcd}$$

- For stable maps g and g' we always have $\Sigma_f(\Delta_g(x)) \cong \Delta_{g'}(\Sigma_{f'}(x))$,
- For exponentiable maps f and f' we always have $\Pi_f(\Delta_g(x)) \cong \Delta_{g'}(\Pi_{f'}(x))$ (follows from above by adjointness).

Composition of polynomials

Want polynomial composition to correspond to polynomial functor composition:

$$
\begin{tikzcd}
 & & & P_2 \arrow[r, "u_2"] \arrow[dl, "e"'] & S_2 \arrow[r, "u_1'"] \arrow[dl, "\varepsilon"'] \arrow[dd] & S_2' \arrow[dd, "w_1"] & \\
 & & P_1 \arrow[r, "u_1"] \arrow[dl, "v_2'"'] & S_1 \arrow[dl, "v_1'"] \arrow[dr, "w_1"] & & & \\
 & P \arrow[r, "u"'] \arrow[dl, "v"] & S \arrow[dr, "w"'] & & P' \arrow[r, "u'"'] \arrow[dl, "v'"] & S' \arrow[dr, "w'"'] & \\
X & & & Y & & & Z
\end{tikzcd}
$$

$$
\begin{aligned}
P_{u',v',w'}(P_{u,v,w}(x)) &= \Sigma_{w'}(\Pi_{u'}(\Delta_{v'}(\Sigma_w(\Pi_u(\Delta_v(x)))))) && \text{BC} \\
&= \Sigma_{w'}(\Pi_{u'}(\Sigma_{w_1}(\Delta_{v_1'}(\Pi_u(\Delta_v(x)))))) && \text{BC} \\
&= \Sigma_{w'}(\Pi_{u'}(\Sigma_{w_1}(\Pi_{u_1}(\Delta_{v_2' v}((x)))))) && \text{lemma} \\
&= \Sigma_{w_1 w'}(\Pi_{u_1'}(\Delta_{\varepsilon}(\Pi_{u_1}(\Delta_{v_2' v}((x)))))) && \text{BC} \\
&= \Sigma_{w_1 w'}(\Pi_{u_2 u_1}(\Delta_{e v_2' v}((x))))
\end{aligned}
$$

So composition is associative (up to equivalence).

Morphism of polynomials

$$
\begin{tikzcd}
X \arrow[dd, equal] & P \arrow[l, "v"'] \arrow[r, "u"] & S \arrow[r, "w"'] \arrow[d, equal] & Y \arrow[dd, equal] \\
& \Delta_{u'}(\alpha)_0 \arrow[u, "\beta"] \arrow[r] \arrow[d, "\Delta_{u'}(\alpha)"'] & S \arrow[d, "\alpha"] & \\
X & P' \arrow[l, "v'"'] \arrow[r, "u'"] & S' \arrow[r, "w'"'] & Y
\end{tikzcd}
$$

When β is an isomorphism the morphism of polynomials is **Cartesian**.

Morphism of polynomials

The Cartesian part ...

$$\begin{tikzcd}
X \arrow[d, equal] & V \arrow[l, "v"'] \arrow[r, "u"] \arrow[d, "\alpha"'] & S \arrow[r, "w"] \arrow[d, "\beta"] & Y \arrow[d, equal] \\
X & V' \arrow[l, "v'"] \arrow[r, "u'"'] & S' \arrow[r, "w'"'] & Y
\end{tikzcd}$$

Gives a Cartesian strong natural transformation between polynomial functors:

$$\begin{tikzcd}
\mathbb{X}/X \arrow[rr, "\Delta_v"] \arrow[dr, "\Delta_{v'}"'] & & \mathbb{X}/V \arrow[rr, "\Pi_u"] & & \mathbb{X}/S \arrow[rr, "\Sigma_w"] \arrow[dr, "\Sigma_\beta"] \arrow[d, phantom, "\downarrow\epsilon"] & & \mathbb{X}/Y \\
& \mathbb{X}/V' \arrow[ur, "\Delta_\alpha"'] \arrow[rr, "\Pi_{u'}"'] & & \mathbb{X}/S' \arrow[ur, "\Delta_\beta"] \arrow[rr, equal] & & \mathbb{X}/S' \arrow[ur, "\Sigma_{w'}"'] &
\end{tikzcd}$$

Note ϵ is strong Cartesian and all functors preserve pullbacks.

Morphism of polynomials

The rest ...

$$\begin{tikzcd}
X \arrow[d, equal] & V \arrow[l, "v"'] \arrow[r, "u"] & S \arrow[d, equal] \arrow[r, "w"] & Y \arrow[d, equal] \\
X & V' \arrow[l, "v'"] \arrow[r, "u'"'] \arrow[u, "\gamma"] & S \arrow[r, "w"'] & Y
\end{tikzcd}$$

Gives a strong natural transformation between polynomial functors:

$$\begin{tikzcd}
 & \mathbb{X}/V \arrow[rr, equal] \arrow[dr, "\Delta_\gamma"'] & \arrow[d, phantom, "\downarrow\eta"] & \mathbb{X}/V \arrow[dr, "\Pi_u"] & & \\
\mathbb{X}/X \arrow[ur, "\Delta_v"] \arrow[rr, "\Delta_{v'}"'] & & \mathbb{X}/V' \arrow[ur, "\Pi_\gamma"'] \arrow[rr, "\Pi_{u'}"'] & & \mathbb{X}/S' \arrow[r, "\Sigma_w"'] & \mathbb{X}/Y
\end{tikzcd}$$

Note: η is strong but not Cartesian.

Morphism of polynomials

This gives an exact correspondence between strong natural transformations between polynomial functors and morphisms of polynomials.

NOTE: all these natural transformations are generated by η and ϵ ...

The bicategory of polynomials
Clearly polynomials form a bicategory ...

They also naturally form a double category ....

Just mentioned *that* to keep Robert Pare happy!

Polynomials are left additive

The addition is given by coproduct:

$$
\begin{tikzcd}
 & P' \arrow[r, "u'"] \arrow[ddl, "v'"'] & S' \arrow[ddr, "w'"] & \\
 & P \arrow[r, "u"] \arrow[dl, "v"] & S \arrow[dr, "w"'] & \\
X & & & Y
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & P+P' \arrow[r, "u+u'"] \arrow[dl, "\langle v|v' \rangle"] & S+S' \arrow[dr, "\langle w|w' \rangle"'] & \\
X & & & Y
\end{tikzcd}
$$

Most maps *not* additive ... spans are!

Products in the category of polynomials

Here is the pairing operation:

$$
\begin{tikzcd}
 & P' \arrow[r, "u'"] \arrow[ddl, "v'"'] & S' \arrow[dr, "w'"] & \\
 & P \arrow[r, "u"] \arrow[dl, "v"] & S \arrow[dr, "w"'] & Y' \\
X & & & Y
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & P + P' \arrow[r, "u+u'"] \arrow[dl, "\langle v|v' \rangle"] & S + S' \arrow[dr, "w+w'"'] & \\
X & & & Y + Y'
\end{tikzcd}
$$

Given by coproduct ...

Need the category to be extensive.

Where are we?

- Structural polynomials correspond to polynomial functors
- (Strong) natural transformations correspond to morphisms of structural polynomials
- Polynomials form a left additive (bi)category (whatever that is!!!)
- Can express initial datatypes and a lot else besides by polynomials

CAN WE DIFFERENTIATE POLYNOMIALS?

Differentiating polynomials

Say a map u is **separable** when the diagonal in the kernel of u is detachable (i.e. it is a coproduct component). That is the following diagram

$$\begin{tikzcd} P+Q \arrow[r, "\langle 1|q_0\rangle"] \arrow[d, "\langle 1|q_1\rangle"] & P \arrow[d, "u"] \\ P \arrow[r, "u"'] & S \end{tikzcd}$$

is a pullback.

Consider only separable polynomials (i.e. with u separable) ...
Closed to all basic constructions (composition, addition, ...).

Differentiating polynomials

Can differentiate polynomials whose multiplicity assignment u is separable:

$$\dfrac{\begin{tikzcd} & P \arrow[r, "u"] \arrow[dl, "v"] & S \arrow[dr, "w"'] & \\ X & & & Y \end{tikzcd}}{\begin{tikzcd} & P+Q \arrow[r, "\langle 1 | q_1 \rangle"] \arrow[dl, "v + q_0 v"] & P \arrow[dr, "uw"'] & \\ X+X & & & Y \end{tikzcd}}\ \mathrm{Diff}$$

Differentiating polynomials

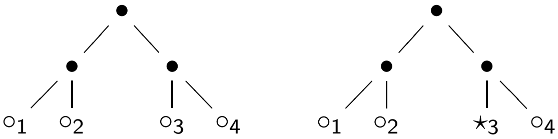
- Can differentiate data types:
  ... agrees with existing notion
  (in fact, clarifies notion somewhat).
- Can differentiate combinatorial species:
  ... agrees with existing notion (for polynomial functors).
- An example of a differential category in which negation is unnatural!
- Of course, need to prove this is a differential!!!
  (chain rule is already challenging ...)

Differentiating polynomials

SO WHAT IS THE DIFFERENTIAL OF A TREE!

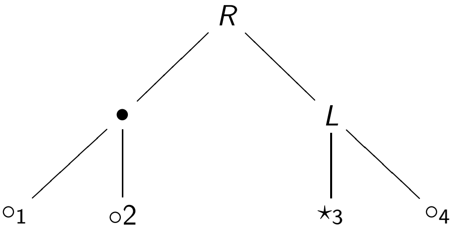
Differentiating polynomials

Essentially it is a tree with a leaf picked out and given a new variable name ...



Differentiating polynomials

But how do you express this as a data type?



Differentiating polynomials

–Tree data type based on product

```
data Prod a = Prod a a
data Tree a = Node Prod (Tree a)
              Leaf a
```

– Differential of tree based on differential of product ...

```
data DProd b a = R b a
                 L a b
data DTree b a = DNode (DProd (DTree b a) (Tree a) )
                 DLeaf b
```

Concluding remarks

- Is this all completely sorted out?
  Absolutely not!
  (BUT there is a lot there already!)
- Are polynomial functors the only ones which can be differentiated?
  Certainly not: just an important class!
- Is *this* example of a differential useful?
  Amazingly the answer is probably YES!!!

END

Some references

[1] N. Gambino and J. Kock,
Polynomial Functors and Polynomial Monads
ArXiv:0906.4931 (2010)

[2] T. Altenkirch and P. Morris, *Indexed containers*
In LICS'09 277-285 (2009)

[3] M. Abbott, T. Altenkirch, C. McBride, and N. Gahni,
Differentiating data structures, Fund. Inf. 65(1-2):1-28 (2005)

[4] G. Huet, Functional Pearl: *The zipper.* Journal of Functional programming (5):549-554 (1997)

[5] N. Gambino and M. Hyland
Wellfounded trees and dependent polynomial functors
In TYPES'03, pages 210-225 (2003)